1. Adobe ColdFusion CFML Reference
   1.1 CFML Reference (ColdFusion 11) ................................................................. 19
   1.2 Reserved Words and Variables ...................................................................... 20
       1.2.1 Reserved words ....................................................................................... 21
           1.2.1.1 Reserved words in forms ................................................................. 22
           1.2.1.2 Reserved words in queries ............................................................... 23
       1.2.2 Scope-specific built-in variables ............................................................... 24
           1.2.2.1 Variable scope ................................................................................. 25
           1.2.2.2 Caller scope .................................................................................... 26
           1.2.2.3 CGI variables .................................................................................. 27
           1.2.2.4 Client variables ............................................................................... 28
           1.2.2.5 Server variables ............................................................................. 29
       1.2.3 Application and session variables .............................................................. 30
           1.2.3.1 Request variable .............................................................................. 31
           1.2.3.2 Form variable .................................................................................. 32
       1.2.4 ColdFusion tag-specific variables .............................................................. 33
           1.2.4.1 ColdFusion query variables .............................................................. 34
           1.2.4.2 CFCATCH variables ....................................................................... 35
           1.2.4.3 CFDIRECTORY variables ................................................................. 36
           1.2.4.4 CFERROR variables ....................................................................... 37
           1.2.4.5 CF_FILE_ACTION=Upload variables .............................................. 38
           1.2.4.6 CFFTP error variables ................................................................... 39
           1.2.4.7 CFFTP ReturnValue variable ........................................................... 40
           1.2.4.8 CFFTP query object columns ........................................................... 41
           1.2.4.9 CFHTTP variables ......................................................................... 42
           1.2.4.10 CFLDAP variables ....................................................................... 43
           1.2.4.11 CFPOP variables ......................................................................... 44
           1.2.4.12 CFQUERY and CFSTOREDPROC variables .................................. 45
           1.2.4.13 CFREGISTRY variables .................................................................. 46
           1.2.4.14 CFSEARCH variables .................................................................. 47
       1.2.5 CGI environment (CGI Scope) variables .................................................. 48
           1.2.5.1 Testing for CGI variables ................................................................ 49
           1.2.5.2 CGI server variables ..................................................................... 50
           1.2.5.3 CGI client variables ...................................................................... 51
           1.2.5.4 CGI client certificate variables ....................................................... 52
   1.3 ColdFusion Tags
       1.3.1 Tags in ColdFusion 10 ............................................................................ 53
       1.3.2 Tag summary ......................................................................................... 54
       1.3.3 Tags by function
           1.3.3.1 Application framework tags .............................................................. 55
           1.3.3.2 Communications tags .................................................................... 56
           1.3.3.3 Database manipulation tags ............................................................. 57
           1.3.3.4 Data output tags ............................................................................ 58
           1.3.3.5 Debugging tags .............................................................................. 59
           1.3.3.6 Display management tags ............................................................... 60
           1.3.3.7 Exception handling tags ................................................................. 61
           1.3.3.8 Extensibility tags ........................................................................... 62
           1.3.3.9 File management tags ................................................................... 63
           1.3.3.10 Flow-control tags ....................................................................... 64
           1.3.3.11 Forms tags .................................................................................. 65
           1.3.3.12 Internet protocol tags ................................................................... 66
           1.3.3.13 Page processing tags ................................................................... 67
           1.3.3.14 Security tags ............................................................................... 68
           1.3.3.15 Variable manipulation tags ............................................................ 69
           1.3.3.16 Web Socket tags ......................................................................... 70
           1.3.3.17 Other tags ................................................................................... 71
       1.3.4 Tag changes since ColdFusion 5 ................................................................. 72
           1.3.4.1 New tags, attributes, and values ....................................................... 73
           1.3.4.2 Deprecated tags, attributes, and values ......................................... 74
           1.3.4.3 Obsolete tags, attributes, and values ............................................. 75
       1.3.5 Tags a-b
           1.3.5.1 cbabort ........................................................................................... 76
           1.3.5.2 cfajaximport ............................................................................... 77
           1.3.5.3 cfajaxproxy .................................................................................. 78
           1.3.5.4 cfcookie ....................................................................................... 79
           1.3.5.5 cfappication ............................................................................... 80
           1.3.5.6 cflagument .................................................................................. 81
           1.3.5.7 cfassociate .................................................................................. 82
           1.3.5.8 cfarray ......................................................................................... 83
           1.3.5.9 cfarraylist .................................................................................... 84
           1.3.5.10 cfbreak ........................................................................................ 85
           1.3.5.11 cfcatch ....................................................................................... 86
           1.3.5.12 cfcatcherror .............................................................................. 87
           1.3.5.13 cfcatcherrorrange ..................................................................... 88
           1.3.5.14 cfcatcherrorvalue ..................................................................... 89
           1.3.5.15 cfcatcherrorvar ........................................................................ 90
           1.3.5.16 cfcatcherrorvarrange ................................................................. 91
           1.3.5.17 cfcatcherrorvarvalue ................................................................. 92
           1.3.5.18 cfcatcherrorvarvalue ................................................................ 93
           1.3.5.19 cfcatcherrorvarvalue ................................................................ 94
           1.3.5.20 cfcatcherrorvarvalue ................................................................ 95
           1.3.5.21 cfcatcherrorvarvalue ................................................................ 96
           1.3.5.22 cfcatcherrorvarvalue ................................................................ 97
           1.3.5.23 cfcatcherrorvarvalue ................................................................ 98
           1.3.5.24 cfcatcherrorvarvalue ................................................................ 99
           1.3.5.25 cfcatcherrorvarvalue ................................................................. 100
           1.3.5.26 cfcatcherrorvarvalue ................................................................. 101
           1.3.5.27 cfcatcherrorvarvalue ................................................................. 102
           1.3.5.28 cfcatcherrorvarvalue ................................................................. 103
           1.3.5.29 cfcatcherrorvarvalue ................................................................. 104
           1.3.5.30 cfcatcherrorvarvalue ................................................................. 105
           1.3.5.31 cfcatcherrorvarvalue ................................................................. 106
           1.3.5.32 cfcatcherrorvarvalue ................................................................. 107
1.3.5.8 cfauthorize
1.3.5.9 cfbreak
1.3.6 Tags c
1.3.6.1 cfcache
1.3.6.2 cfcalendar
1.3.6.3 cfcase
1.3.6.4 cfcatch
1.3.6.5 cfchart
1.3.6.6 cfchartdata
1.3.6.7 cfchartseries
1.3.6.8 cfcol
1.3.6.9 cfcollection
1.3.6.10 cfcomponent
1.3.6.11 cfcontent
1.3.6.12 cfcontinue
1.3.6.13 cfcookie
1.3.6.14 cftcclient
1.3.6.15 cftcclientsettings
1.3.7 Tags d-e
1.3.7.1 cfdbinfo
1.3.7.2 cfdefaultcase
1.3.7.3 cdirectory
1.3.7.4 cfdiv
1.3.7.5 cdocumento
1.3.7.6 cdocumentoitem
1.3.7.7 cdocumentoaction
1.3.7.8 cfdump
1.3.7.9 cfelse
1.3.7.10 cfelseif
1.3.7.11 cferror
1.3.7.12 cfexchangelocal
1.3.7.13 cfexchangeremote
1.3.7.14 cfexchangecontact
1.3.7.15 cfexchangemail
1.3.7.16 cfexchangefilter
1.3.7.17 cfexchangelocal
1.3.7.18 cfexchangecontent
1.3.7.19 cfexchangeconnection
1.3.7.20 cfexecute
1.3.7.21 cfexit
1.3.8 Tags f
1.3.8.1 cffeed
1.3.8.2 cfif
1.3.8.3 cfif action = "append"
1.3.8.4 cfif action = "copy"
1.3.8.5 cfif action = "delete"
1.3.8.6 cfif action = "move"
1.3.8.7 cfif action = "read"
1.3.8.8 cfif action = "readBinary"
1.3.8.9 cfif action = "rename"
1.3.8.10 cfif action = "upload"
1.3.8.11 cfif action = "uploadAll"
1.3.8.12 cfif action = "write"
1.3.8.13 cfifupload
1.3.8.14 cfiffinally
1.3.8.15 cfifflush
1.3.8.16 cfifgroup
1.3.8.17 cfifitem
1.3.8.18 cfif
1.3.8.19 cftp
1.3.8.19.1 cftp: Opening and closing FTP server connections
1.3.8.20 cftp
1.3.8.21 cftp: Connection: file and directory operations
1.3.8.22 cftp action = "listDir"
1.3.8.23 cftpmodule
1.3.8.24 cfmodule
1.3.9 Tags g-h
1.3.9.1 cfgraph
1.3.9.2 cfgraphdata
1.3.9.3 cfgrid
1.3.9.4 cfgridcolumn
1.3.9.5 cfgridrow
1.3.9.6 cfgridupdate
1.3.13.13 cfprocessingdirective .................................................. 101
1.3.13.14 cfprocparam .............................................................. 102
1.3.13.15 cfprocresult ............................................................... 102
1.3.13.16 cfprogressbar ............................................................. 102
1.3.13.17 cfproperty ................................................................. 103
1.3.13.18 cfquery ................................................................. 104.
1.3.13.19 cfqueryparam ............................................................ 105
1.3.14 Tags r-s ........................................................................ 105
  1.3.14.1 cfregistry ................................................................. 105
  1.3.14.2 cfreport ................................................................. 106
  1.3.14.3 cfreportparam .......................................................... 108
  1.3.14.4 cfthrow ................................................................. 109
  1.3.14.5 cftern ................................................................. 109
  1.3.14.6 cfsavecontent .......................................................... 109
  1.3.14.7 cfschedule ............................................................. 109
  1.3.14.8 cfscript ................................................................. 110
  1.3.14.9 cfsave ................................................................. 111
  1.3.14.10 cselect ................................................................. 112
  1.3.14.11 cfset ................................................................. 113
  1.3.14.12 cfsetparam ........................................................... 113
  1.3.14.13 cfsetting .............................................................. 113
  1.3.14.14 cfsetting .............................................................. 113
  1.3.14.15 cfsharepoint .......................................................... 114
  1.3.14.16 cfsilent ................................................................. 114
  1.3.14.17 cfslider ................................................................. 115
  1.3.14.18 cfspreadsheet ........................................................ 115
  1.3.14.19 cfproperty .......................................................... 116
  1.3.14.20 cfspref ................................................................. 116
  1.3.14.21 cfswitch ............................................................. 117
1.3.15 Tags t ........................................................................... 117
  1.3.15.1 cftable ................................................................. 117
  1.3.15.2 cftextarea .............................................................. 118
  1.3.15.3 cftextarea .............................................................. 119
  1.3.15.4 cfthread ................................................................. 119
  1.3.15.5 cftable ................................................................. 120
  1.3.15.6 cfthread ................................................................. 120
  1.3.15.7 cftemplate ............................................................. 121
  1.3.15.8 cftrace ................................................................. 121
  1.3.15.9 cftransaction .......................................................... 121
  1.3.15.10 cftree ................................................................. 122
  1.3.15.11 cfview ................................................................. 123
  1.3.15.12 cftry ................................................................. 124
1.3.16 Tags u-z ...................................................................... 124
  1.3.16.1 cfupdate ................................................................. 124
  1.3.16.2 cfwindow ............................................................... 125
  1.3.16.3 cfwebsocket ............................................................ 125
  1.3.16.4 cfwindow ............................................................... 125
  1.3.16.5 cfxml ................................................................. 126
  1.3.16.6 cfzip ................................................................. 127
  1.3.16.7 cfzipparam ............................................................ 128
1.4 ColdFusion Functions .................................................................. 129
  1.4.1 New Functions in ColdFusion 10 ........................................ 129
  1.4.2 Functions by category ........................................................ 129
    1.4.2.1 Array functions ....................................................... 129
    1.4.2.2 Cache functions ...................................................... 129
    1.4.2.3 Conversion functions ............................................... 129
    1.4.2.4 Date and time functions .......................................... 130
    1.4.2.5 Data output functions ............................................. 130
    1.4.2.6 Debugging functions .............................................. 130
    1.4.2.7 Decision functions ............................................... 130
    1.4.2.8 Display and formatting functions ............................. 130
    1.4.2.9 Dynamic evaluation functions ................................ 130
    1.4.2.10 Exception handling functions ............................... 130
    1.4.2.11 Extensibility functions ......................................... 130
    1.4.2.12 Flow control functions ........................................ 130
    1.4.2.13 Full-text search functions .................................... 130
    1.4.2.14 Image functions .................................................. 131
    1.4.2.15 International functions ......................................... 131
    1.4.2.16 List functions ..................................................... 131
    1.4.2.17 Mathematical functions ....................................... 131
    1.4.2.18 Microsoft office integration functions .................... 131
    1.4.2.19 ORM functions .................................................... 131
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4.2.20</td>
<td>Other functions</td>
<td>131</td>
</tr>
<tr>
<td>1.4.2.21</td>
<td>Query functions</td>
<td>131</td>
</tr>
<tr>
<td>1.4.2.22</td>
<td>Security functions</td>
<td>131</td>
</tr>
<tr>
<td>1.4.2.23</td>
<td>Spreadsheet functions</td>
<td>131</td>
</tr>
<tr>
<td>1.4.2.24</td>
<td>String functions</td>
<td>132</td>
</tr>
<tr>
<td>1.4.2.25</td>
<td>Structure functions</td>
<td>132</td>
</tr>
<tr>
<td>1.4.2.26</td>
<td>System functions</td>
<td>132</td>
</tr>
<tr>
<td>1.4.2.27</td>
<td>Transaction functions</td>
<td>132</td>
</tr>
<tr>
<td>1.4.2.28</td>
<td>XML functions</td>
<td>132</td>
</tr>
<tr>
<td>1.4.2.29</td>
<td>Mobile Functions</td>
<td>132</td>
</tr>
<tr>
<td>1.4.4.3</td>
<td>Function changes since ColdFusion 5</td>
<td>132</td>
</tr>
<tr>
<td>1.4.4.3.1</td>
<td>New functions, parameters, and values</td>
<td>132</td>
</tr>
<tr>
<td>1.4.4.3.2</td>
<td>Deprecated functions, parameters, and values</td>
<td>133</td>
</tr>
<tr>
<td>1.4.4.3.3</td>
<td>Obsolete functions, parameters, and values</td>
<td>133</td>
</tr>
<tr>
<td>1.4.4.4</td>
<td>Functions a-b</td>
<td>133</td>
</tr>
<tr>
<td>1.4.4.4.1</td>
<td>Abs</td>
<td>133</td>
</tr>
<tr>
<td>1.4.4.4.2</td>
<td>ACos</td>
<td>134</td>
</tr>
<tr>
<td>1.4.4.4.3</td>
<td>AddSOAPRequestHeader</td>
<td>134</td>
</tr>
<tr>
<td>1.4.4.4.4</td>
<td>AddSOAPResponseHeader</td>
<td>134</td>
</tr>
<tr>
<td>1.4.4.4.5</td>
<td>AjaxLink</td>
<td>134</td>
</tr>
<tr>
<td>1.4.4.4.6</td>
<td>AjaxOnLoad</td>
<td>134</td>
</tr>
<tr>
<td>1.4.4.4.7</td>
<td>ApplicationStop</td>
<td>134</td>
</tr>
<tr>
<td>1.4.4.4.8</td>
<td>ArrayAppend</td>
<td>135</td>
</tr>
<tr>
<td>1.4.4.4.9</td>
<td>ArrayAvg</td>
<td>135</td>
</tr>
<tr>
<td>1.4.4.4.10</td>
<td>ArrayClear</td>
<td>135</td>
</tr>
<tr>
<td>1.4.4.4.11</td>
<td>ArrayContains</td>
<td>135</td>
</tr>
<tr>
<td>1.4.4.4.12</td>
<td>ArrayDelete</td>
<td>135</td>
</tr>
<tr>
<td>1.4.4.4.13</td>
<td>ArrayDeleteAt</td>
<td>136</td>
</tr>
<tr>
<td>1.4.4.4.14</td>
<td>ArrayEach</td>
<td>136</td>
</tr>
<tr>
<td>1.4.4.4.15</td>
<td>ArrayFilter</td>
<td>136</td>
</tr>
<tr>
<td>1.4.4.4.16</td>
<td>ArrayFind</td>
<td>136</td>
</tr>
<tr>
<td>1.4.4.4.17</td>
<td>ArrayFindAll</td>
<td>136</td>
</tr>
<tr>
<td>1.4.4.4.18</td>
<td>ArrayFindAllNoCase</td>
<td>136</td>
</tr>
<tr>
<td>1.4.4.4.19</td>
<td>ArrayFindNoCase</td>
<td>136</td>
</tr>
<tr>
<td>1.4.4.4.20</td>
<td>ArrayInsertAt</td>
<td>137</td>
</tr>
<tr>
<td>1.4.4.4.21</td>
<td>ArrayIsDefined</td>
<td>137</td>
</tr>
<tr>
<td>1.4.4.4.22</td>
<td>ArrayIsEmpty</td>
<td>137</td>
</tr>
<tr>
<td>1.4.4.4.23</td>
<td>arrayLen</td>
<td>137</td>
</tr>
<tr>
<td>1.4.4.4.24</td>
<td>ArrayMap</td>
<td>137</td>
</tr>
<tr>
<td>1.4.4.4.25</td>
<td>ArrayMax</td>
<td>138</td>
</tr>
<tr>
<td>1.4.4.4.26</td>
<td>ArrayMin</td>
<td>138</td>
</tr>
<tr>
<td>1.4.4.4.27</td>
<td>ArrayNew</td>
<td>138</td>
</tr>
<tr>
<td>1.4.4.4.28</td>
<td>ArrayPrepend</td>
<td>138</td>
</tr>
<tr>
<td>1.4.4.4.29</td>
<td>ArrayReduce</td>
<td>138</td>
</tr>
<tr>
<td>1.4.4.4.30</td>
<td>ArrayResize</td>
<td>138</td>
</tr>
<tr>
<td>1.4.4.4.31</td>
<td>ArraySet</td>
<td>138</td>
</tr>
<tr>
<td>1.4.4.4.32</td>
<td>ArraySlice</td>
<td>139</td>
</tr>
<tr>
<td>1.4.4.4.33</td>
<td>ArraySort</td>
<td>139</td>
</tr>
<tr>
<td>1.4.4.4.34</td>
<td>ArraySum</td>
<td>139</td>
</tr>
<tr>
<td>1.4.4.4.35</td>
<td>ArraySwap</td>
<td>139</td>
</tr>
<tr>
<td>1.4.4.4.36</td>
<td>ArrayToList</td>
<td>139</td>
</tr>
<tr>
<td>1.4.4.4.37</td>
<td>Asc</td>
<td>139</td>
</tr>
<tr>
<td>1.4.4.4.38</td>
<td>ASin</td>
<td>140</td>
</tr>
<tr>
<td>1.4.4.4.39</td>
<td>Atn</td>
<td>140</td>
</tr>
<tr>
<td>1.4.4.4.40</td>
<td>AuthenticatedContext</td>
<td>140</td>
</tr>
<tr>
<td>1.4.4.4.41</td>
<td>AuthenticatedUser</td>
<td>140</td>
</tr>
<tr>
<td>1.4.4.4.42</td>
<td>BinaryDecode</td>
<td>140</td>
</tr>
<tr>
<td>1.4.4.4.43</td>
<td>BinaryEncode</td>
<td>140</td>
</tr>
<tr>
<td>1.4.4.4.44</td>
<td>BitAnd</td>
<td>141</td>
</tr>
<tr>
<td>1.4.4.4.45</td>
<td>BitMaskClear</td>
<td>141</td>
</tr>
<tr>
<td>1.4.4.4.46</td>
<td>BitMaskRead</td>
<td>141</td>
</tr>
<tr>
<td>1.4.4.4.47</td>
<td>BitMaskSet</td>
<td>141</td>
</tr>
<tr>
<td>1.4.4.4.48</td>
<td>BitNot</td>
<td>141</td>
</tr>
<tr>
<td>1.4.4.4.49</td>
<td>BitOr</td>
<td>141</td>
</tr>
<tr>
<td>1.4.4.50</td>
<td>BitSHL</td>
<td>141</td>
</tr>
<tr>
<td>1.4.4.51</td>
<td>BitSHRN</td>
<td>141</td>
</tr>
<tr>
<td>1.4.4.52</td>
<td>BitXor</td>
<td>141</td>
</tr>
<tr>
<td>1.4.4.53</td>
<td>ArrayDeleteNoCase</td>
<td>142</td>
</tr>
<tr>
<td>1.4.5.1</td>
<td>Functions c-d</td>
<td>142</td>
</tr>
<tr>
<td>1.4.5.1.1</td>
<td>CacheDelete</td>
<td>142</td>
</tr>
<tr>
<td>1.4.5.1.2</td>
<td>CacheGet</td>
<td>142</td>
</tr>
<tr>
<td>1.4.5.1.3</td>
<td>CacheGetAllIds</td>
<td>142</td>
</tr>
</tbody>
</table>
1.4.5.76 DeserializeXML ................................................................. 158
1.4.6 Functions e-g ........................................................................ 158
1.4.6.1 EncodeForCSS .................................................................... 158
1.4.6.2 EncodeForHTML ................................................................. 158
1.4.6.3 EncodeForHTMLAttribute .................................................. 158
1.4.6.4 EncodeForJavaScript .......................................................... 159
1.4.6.5 EncodeForURL .................................................................... 159
1.4.6.6 EncodeForXML ................................................................... 159
1.4.6.7 Encrypt .............................................................................. 159
1.4.6.8 GetTempDirectory .............................................................. 160
1.4.6.9 GetApplicationMetadata ..................................................... 160
1.4.6.10 EncryptBinary ................................................................... 160
1.4.6.11 EntityDelete ...................................................................... 161
1.4.6.12 EntityLoad ........................................................................ 161
1.4.6.13 EntityLoadByExample ......................................................... 161
1.4.6.14 EntityLoadByPK ................................................................ 161
1.4.6.15 EntityMerge ...................................................................... 161
1.4.6.16 EntityNew ......................................................................... 161
1.4.6.17 EntityReload ...................................................................... 161
1.4.6.18 EntitySave ........................................................................ 162
1.4.6.19 EntityToQuery ................................................................... 162
1.4.6.20 Evaluate ............................................................................ 162
1.4.6.21 Exp .................................................................................. 162
1.4.6.22 ExpandPath ...................................................................... 162
1.4.6.23 FileClose ........................................................................... 163
1.4.6.24 FileCopy ............................................................................ 163
1.4.6.25 FileDelete ......................................................................... 163
1.4.6.26 FileExists ......................................................................... 163
1.4.6.27 FileGetMimeType ............................................................... 163
1.4.6.28 FileIsEOF ......................................................................... 164
1.4.6.29 FileMove ........................................................................... 164
1.4.6.30 FileOpen ........................................................................... 164
1.4.6.31 FileRead ............................................................................ 164
1.4.6.32 FileReadBinary ................................................................. 164
1.4.6.33 FileReadLine ..................................................................... 165
1.4.6.34 FileSeek ........................................................................... 165
1.4.6.35 FileSetAccessMode ........................................................... 165
1.4.6.36 FileSetAttribute ............................................................... 165
1.4.6.37 FileSetLastModified .......................................................... 165
1.4.6.38 FileSkipBytes ................................................................... 165
1.4.6.39 FileUpload ........................................................................ 165
1.4.6.40 FileUploadAll .................................................................... 166
1.4.6.41 FileWriter ......................................................................... 166
1.4.6.42 FileWriter ......................................................................... 166
1.4.6.43 Find .................................................................................. 166
1.4.6.44 FindNoCase ...................................................................... 167
1.4.6.45 FindOneOf ........................................................................ 167
1.4.6.46 FirstDayOfMonth ............................................................... 167
1.4.6.47 Fix ................................................................................... 167
1.4.6.48 FormatBaseN .................................................................... 167
1.4.6.49 GenerateSecretKey ............................................................. 167
1.4.6.50 GetAuthUser ..................................................................... 168
1.4.6.51 GetBaseTagData ................................................................. 168
1.4.6.52 GetBaseTagList .................................................................. 168
1.4.6.53 GetBaseTemplatePath ........................................................ 168
1.4.6.54 GetClientVariablesList ...................................................... 168
1.4.6.55 GetComponentMetaData ...................................................... 169
1.4.6.56 GetContextRoot ................................................................. 169
1.4.6.57 GetCPUUsage ................................................................. 169
1.4.6.58 GetCurrentTemplatePath .................................................... 169
1.4.6.59 GetDirectoryFromPath ....................................................... 169
1.4.6.60 GetEncoding ..................................................................... 169
1.4.6.61 GetException ..................................................................... 169
1.4.6.62 GetFromFilePath ............................................................... 169
1.4.6.63 GetFileInfo ........................................................................ 170
1.4.6.64 GetFreeSpace ..................................................................... 170
1.4.6.65 GetFunctionCalledName ..................................................... 170
1.4.6.66 GetFunctionList ............................................................... 170
1.4.6.67 GetGatewayHelper ............................................................ 170
1.4.6.68 GetHttpRequestData .......................................................... 171
1.4.6.69 GetHttpTimeString ............................................................ 171
1.4.6.70 GetK2ServerDocCount ........................................................ 171
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetK2ServerDocCountLimit</td>
<td>171</td>
</tr>
<tr>
<td>GetLocale</td>
<td>171</td>
</tr>
<tr>
<td>GetLocaleDisplayName</td>
<td>171</td>
</tr>
<tr>
<td>GetLocalHostIP</td>
<td>171</td>
</tr>
<tr>
<td>GetMetaData</td>
<td>172</td>
</tr>
<tr>
<td>GetMetricData</td>
<td>172</td>
</tr>
<tr>
<td>GeneratePBKDFKey</td>
<td>172</td>
</tr>
<tr>
<td>GetPageContext</td>
<td>173</td>
</tr>
<tr>
<td>GetPrinterInfo</td>
<td>173</td>
</tr>
<tr>
<td>GetPrinterList</td>
<td>173</td>
</tr>
<tr>
<td>GetProfileSections</td>
<td>173</td>
</tr>
<tr>
<td>GetProfileString</td>
<td>173</td>
</tr>
<tr>
<td>GetReadableImageFormats</td>
<td>173</td>
</tr>
<tr>
<td>GetSOAPRequest</td>
<td>174</td>
</tr>
<tr>
<td>GetSOAPRequestHeader</td>
<td>174</td>
</tr>
<tr>
<td>GetSOAPResponse</td>
<td>174</td>
</tr>
<tr>
<td>GetSOAPResponseHeader</td>
<td>174</td>
</tr>
<tr>
<td>GetSystemFreeMemory</td>
<td>174</td>
</tr>
<tr>
<td>GetSystemTotalMemory</td>
<td>175</td>
</tr>
<tr>
<td>GetTempFile</td>
<td>175</td>
</tr>
<tr>
<td>GetTemplatePath</td>
<td>175</td>
</tr>
<tr>
<td>GetTickCount</td>
<td>175</td>
</tr>
<tr>
<td>GetTimeZoneInfo</td>
<td>175</td>
</tr>
<tr>
<td>GetToken</td>
<td>175</td>
</tr>
<tr>
<td>GetTotalSpace</td>
<td>176</td>
</tr>
<tr>
<td>GetUserRoles</td>
<td>176</td>
</tr>
<tr>
<td>GetVFSMetadata</td>
<td>176</td>
</tr>
<tr>
<td>GetWriteableImageFormats</td>
<td>176</td>
</tr>
<tr>
<td>GetSafeHTML</td>
<td>176</td>
</tr>
<tr>
<td>EncodeForXMLAttribute</td>
<td>177</td>
</tr>
<tr>
<td>EncodeForXpath</td>
<td>177</td>
</tr>
<tr>
<td>Hash</td>
<td>177</td>
</tr>
<tr>
<td>HMAC</td>
<td>177</td>
</tr>
<tr>
<td>HQLMethods</td>
<td>178</td>
</tr>
<tr>
<td>Hour</td>
<td>178</td>
</tr>
<tr>
<td>HTMLCodeFormat</td>
<td>178</td>
</tr>
<tr>
<td>HTMLEditFormat</td>
<td>178</td>
</tr>
<tr>
<td>Ill</td>
<td>178</td>
</tr>
<tr>
<td>ImageBlur</td>
<td>179</td>
</tr>
<tr>
<td>ImageCreateCaptcha</td>
<td>179</td>
</tr>
<tr>
<td>ImageCopy</td>
<td>179</td>
</tr>
<tr>
<td>ImageCreate</td>
<td>180</td>
</tr>
<tr>
<td>ImageCrop</td>
<td>180</td>
</tr>
<tr>
<td>ImageDrawArc</td>
<td>180</td>
</tr>
<tr>
<td>ImageDrawBeveledRect</td>
<td>180</td>
</tr>
<tr>
<td>ImageDrawCubicCurve</td>
<td>180</td>
</tr>
<tr>
<td>ImageDrawLine</td>
<td>181</td>
</tr>
<tr>
<td>ImageDrawLines</td>
<td>181</td>
</tr>
<tr>
<td>ImageDrawOval</td>
<td>181</td>
</tr>
<tr>
<td>ImageDrawPoint</td>
<td>181</td>
</tr>
<tr>
<td>ImageDrawQuadraticCurve</td>
<td>181</td>
</tr>
<tr>
<td>ImageDrawRect</td>
<td>182</td>
</tr>
<tr>
<td>ImageDrawRoundRect</td>
<td>182</td>
</tr>
<tr>
<td>ImageDrawText</td>
<td>182</td>
</tr>
<tr>
<td>ImageFlip</td>
<td>182</td>
</tr>
<tr>
<td>ImageGetBlob</td>
<td>183</td>
</tr>
<tr>
<td>ImageGetBufferedImage</td>
<td>183</td>
</tr>
<tr>
<td>ImageGetEXIFMetadata</td>
<td>183</td>
</tr>
<tr>
<td>ImageGetEXIFTag</td>
<td>183</td>
</tr>
<tr>
<td>ImageGetHeight</td>
<td>183</td>
</tr>
<tr>
<td>ImageGetPTCMetadata</td>
<td>184</td>
</tr>
<tr>
<td>ImageGetPTCTag</td>
<td>184</td>
</tr>
<tr>
<td>ImageGetWidth</td>
<td>184</td>
</tr>
<tr>
<td>ImageGrayscale</td>
<td>184</td>
</tr>
<tr>
<td>ImageInfo</td>
<td>184</td>
</tr>
<tr>
<td>ImageMakeColorTransparent</td>
<td>184</td>
</tr>
<tr>
<td>ImageMakeTranslucent</td>
<td>184</td>
</tr>
<tr>
<td>ImageNegative</td>
<td>185</td>
</tr>
<tr>
<td>ImageNew</td>
<td>185</td>
</tr>
<tr>
<td>ImageOverlay</td>
<td>185</td>
</tr>
</tbody>
</table>
1.4.7.41 ImagePaste
1.4.7.42 ImageRead
1.4.7.43 ImageReadBase64
1.4.7.44 ImageResize
1.4.7.45 ImageRotate
1.4.7.46 ImageRotateDrawingAxis
1.4.7.47 ImageScaleToFit
1.4.7.48 ImageSetAntialiasing
1.4.7.49 ImageSetBackgroundColor
1.4.7.50 ImageSetDrawingColor
1.4.7.51 ImageSetDrawingStroke
1.4.7.52 ImageSetDrawingTransparency
1.4.7.53 ImageSharpen
1.4.7.54 ImageShear
1.4.7.55 ImageShearDrawingAxis
1.4.7.56 ImageTranslate
1.4.7.57 ImageTranslateDrawingAxis
1.4.7.58 ImageWrite
1.4.7.59 ImageWriteBase64
1.4.7.60 ImageXORDrawingMode
1.4.7.61 ImageGetDataMetadata

1.4.8 Functions i-k
1.4.8.1 IncrementValue
1.4.8.2 InputBaseN
1.4.8.3 Insert
1.4.8.4 Int
1.4.8.5 Invoke
1.4.8.6 IsArray
1.4.8.7 IsAuthenticated
1.4.8.8 IsAuthorized
1.4.8.9 IsBinary
1.4.8.10 IsBoolean
1.4.8.11 IsClosure
1.4.8.12 IsCustomFunction
1.4.8.13 IsDate
1.4.8.14 IsDDX
1.4.8.15 IsDebugMode
1.4.8.16 IsDefined
1.4.8.17 IsImage
1.4.8.18 IsImageFile
1.4.8.19 IsInstanceOf
1.4.8.20 IsIPv6
1.4.8.21 IsJSON
1.4.8.22 IsK2ServerABroker
1.4.8.23 IsK2ServerDocCountExceeded
1.4.8.24 IsK2ServerOnline
1.4.8.25 IsLeapYear
1.4.8.26 IsLocalHost
1.4.8.27 IsNull
1.4.8.28 IsNumeric
1.4.8.29 IsNumericDate
1.4.8.30 IsObject
1.4.8.31 IsOnline
1.4.8.32 IsPDFFile
1.4.8.33 IsPDFObject
1.4.8.34 IsProtected
1.4.8.35 IsQuery
1.4.8.36 IsSimpleValue
1.4.8.37 IsSOAPRequest
1.4.8.38 IsSpreadsheetFile
1.4.8.39 IsSpreadsheetObject
1.4.8.40 IsStruct
1.4.8.41 IsUserInAnyRole
1.4.8.42 IsUserInRole
1.4.8.43 IsUserLoggedIn
1.4.8.44 IsValid
1.4.8.45 IsWDDX
1.4.8.46 IsXML
1.4.8.47 IsXMLAttribute
1.4.8.48 IsXMLDoc
1.4.8.49 IsXMLElement
1.4.8.50 IsXMNode
1.4.8.51 IsXmlRoot ................................................................. 198
1.4.8.52 JavaCast ................................................................. 198
1.4.8.53 JStringFormat ......................................................... 199
1.4.8.54 IsSafeHTML ............................................................. 199
1.4.8.55 InvalidateOauthAccessstoken ..................................... 199
1.4.8.56 IsValidOauthAccessstoken ........................................ 199
1.4.8.57 InvokeCFClientFunction ........................................... 199
1.4.9 Functions I ................................................................. 199
1.4.9.1 LCase ................................................................. 199
1.4.9.2 Left ................................................................. 200
1.4.9.3 Len ................................................................. 200
1.4.9.4 ListAppend ............................................................ 200
1.4.9.5 ListChangeDelims .................................................. 200
1.4.9.6 ListContains ......................................................... 200
1.4.9.7 ListContainsNoCase ................................................ 201
1.4.9.8 ListDeleteAt .......................................................... 201
1.4.9.9 ListFilter ............................................................ 201
1.4.9.10 ListFind ............................................................ 201
1.4.9.11 ListFindNoCase ..................................................... 201
1.4.9.12 ListFirst ............................................................ 201
1.4.9.13 ListGetAt ........................................................... 202
1.4.9.14 ListInsertAt ........................................................ 202
1.4.9.15 ListLast ............................................................ 202
1.4.9.16 ListLen .............................................................. 202
1.4.9.17 ListPrepend ........................................................ 202
1.4.9.18 ListQualify .......................................................... 203
1.4.9.19 ListRemoveDuplicates ........................................... 203
1.4.9.20 ListRest .............................................................. 203
1.4.9.21 ListSetAt ........................................................... 203
1.4.9.22 ListSort ............................................................ 203
1.4.9.23 ListToArray .......................................................... 204
1.4.9.24 ListValueCount ..................................................... 204
1.4.9.25 ListValueCountNoCase ........................................... 204
1.4.9.26 LJustify .............................................................. 204
1.4.9.27 Location ............................................................. 204
1.4.9.28 Log ................................................................. 205
1.4.9.29 Log10 ............................................................... 205
1.4.9.30 LSCurrencyFormat ............................................... 205
1.4.9.31 LDateStr ......................................................... 205
1.4.9.32 LDateFormat ........................................................ 206
1.4.9.33 LSEuropeCurrencyFormat ....................................... 206
1.4.9.34 LSisCurrency ....................................................... 206
1.4.9.35 LSisDate ............................................................. 206
1.4.9.36 LSisNumeric ........................................................ 207
1.4.9.37 LSNumberFormat ............................................... 207
1.4.9.38 LSParseCurrency .................................................. 207
1.4.9.39 LSParseDateTime ................................................ 207
1.4.9.40 LSParseEuroCurrency ............................................ 208
1.4.9.41 LSParseNumber .................................................... 208
1.4.9.42 LSTimeFormat ..................................................... 208
1.4.9.43 LTrim ............................................................... 208
1.4.9.44 ListEach ........................................................... 209
1.4.9.45 ListReduce .......................................................... 209
1.4.9.46 ListMap ............................................................. 209
1.4.10 Functions m-r .......................................................... 209
1.4.10.1 Max ................................................................. 209
1.4.10.2 Mid ................................................................. 209
1.4.10.3 Min ................................................................. 209
1.4.10.4 Minute .............................................................. 210
1.4.10.5 Month ............................................................... 210
1.4.10.6 MonthAsString ................................................... 210
1.4.10.7 Now ................................................................. 210
1.4.10.8 NumberFormat ..................................................... 210
1.4.10.9 ObjectEquals ....................................................... 210
1.4.10.10 ObjectLoad .......................................................... 211
1.4.10.11 ObjectSave ......................................................... 211
1.4.10.12 OnWSAuthenticate ................................................. 211
1.4.10.13 ORMCloseSession ................................................. 211
1.4.10.14 ORMCloseAllSessions ............................................ 211
1.4.10.15 ORMClearSession ............................................... 212
1.4.10.16 ORMEvictCollection ............................................. 212
1.4.10.17 ORMEvictEntity .................................................. 212
1.4.11.21 SpreadsheetAddRows .................................. 225
1.4.11.22 SpreadsheetAddSplitPane .................................. 226
1.4.11.23 SpreadsheetCreateSheet .................................. 226
1.4.11.24 SpreadsheetDeleteColumn ................................ 226
1.4.11.25 SpreadsheetDeleteColumns ................................ 226
1.4.11.26 SpreadsheetDeleteRow .................................. 226
1.4.11.27 SpreadsheetDeleteRows .................................. 227
1.4.11.28 SpreadsheetFormatCell .................................. 227
1.4.11.29 SpreadsheetFormatColumn .................................. 227
1.4.11.30 SpreadsheetFormatCellRange .................................. 228
1.4.11.31 SpreadsheetFormatColumns ................................ 228
1.4.11.32 SpreadsheetFormatRow .................................. 228
1.4.11.33 SpreadsheetFormatRows ................................ 228
1.4.11.34 SpreadsheetGetCellComment ................................ 228
1.4.11.35 SpreadsheetGetCellFormula ................................ 229
1.4.11.36 SpreadsheetGetCellValue .................................. 229
1.4.11.37 SpreadsheetInfo ........................................ 229
1.4.11.38 SpreadsheetMergeCells ................................ 229
1.4.11.39 SpreadsheetNew ........................................ 229
1.4.11.40 SpreadsheetRead ........................................ 230
1.4.11.41 SpreadsheetReadBinary .................................. 231
1.4.11.42 SpreadsheetRemoveSheet .................................. 230
1.4.11.43 SpreadsheetSetActiveSheet .................................. 230
1.4.11.44 SpreadsheetSetActiveSheetNumber .......................... 230
1.4.11.45 SpreadsheetSetCellComment .................................. 230
1.4.11.46 SpreadsheetSetCellFormula ................................ 230
1.4.11.47 SpreadsheetSetCellValue .................................. 231
1.4.11.48 SpreadsheetSetColumnWidth ................................ 231
1.4.11.49 SpreadsheetSetColumn .................................. 232
1.4.11.50 SpreadsheetSetColumnWidth .................................. 232
1.4.11.51 SpreadsheetSetRowHeight .................................. 232
1.4.11.52 SpreadsheetShiftColumns .................................. 232
1.4.11.53 SpreadsheetShiftRows .................................. 232
1.4.11.54 SpreadsheetWrite ....................................... 232
1.4.11.55 Sqr ........................................ 233
1.4.11.56 StripCR ........................................ 233
1.4.11.57 StructAppend ........................................ 233
1.4.11.58 StructClear ........................................ 233
1.4.11.59 StructCopy ........................................ 233
1.4.11.60 StructCount ........................................ 234
1.4.11.61 StructDelete ........................................ 234
1.4.11.62 StructEach ........................................ 234
1.4.11.63 StructFilter ........................................ 234
1.4.11.64 StructFind ........................................ 234
1.4.11.65 StructFindKey ........................................ 235
1.4.11.66 StructFindValue .................................... 235
1.4.11.67 StructGet ........................................ 235
1.4.11.68 StructInsert ........................................ 235
1.4.11.69 StructIsEmpty ..................................... 235
1.4.11.70 StructKeyArray .................................... 236
1.4.11.71 StructKeyExists ..................................... 236
1.4.11.72 StructKeyList .................................... 236
1.4.11.73 StructNew ........................................ 236
1.4.11.74 StructSort ....................................... 237
1.4.11.75 StructUpdate ....................................... 237
1.4.11.76 SpreadsheetAddPagebreaks .................................. 238
1.4.11.77 SpreadsheetAddAutoFilter .................................. 238
1.4.11.78 StructReduce ....................................... 237
1.4.11.79 StructMap ....................................... 238
1.4.11.80 Serialize ....................................... 238
1.4.11.81 SerializeXML .................................... 238

1.4.12 Functions t-z ........................................ 238
1.4.12.1 Tan ....................................... 238
1.4.12.2 ThreadJoin .................................... 238
1.4.12.3 ThreadTerminate ................................ 238
1.4.12.4 Throw ....................................... 239
1.4.12.5 TimeFormat ................................... 239
1.4.12.6 ToBase64 ................................... 239
1.4.12.7 ToBinary ................................... 239
1.4.12.8 ToScript ................................... 239
1.4.12.9 ToString ................................... 240
1.4.12.10 Trace ....................................... 240
1.4.12.11 Transaction and concurrency .................................................. 240
1.4.12.12 Transaction Commit .............................................................. 241
1.4.12.13 Transaction Rollback ............................................................. 241
1.4.12.14 Transaction Set SavePoint ..................................................... 241
1.4.12.15 Trim ..................................................................................... 241
1.4.12.16 UCase ................................................................................ 241
1.4.12.17 URLDecode ......................................................................... 241
1.4.12.18 URLEncodedFormat ............................................................... 241
1.4.12.19 URLSessionFormat ............................................................... 241
1.4.12.20 Val ....................................................................................... 242
1.4.12.21 ValueList ............................................................................. 242
1.4.12.22 VerifyClient ....................................................................... 242
1.4.12.23 Week ................................................................................. 242
1.4.12.24 Wrap ................................................................................... 243
1.4.12.25 WriteDump ......................................................................... 243
1.4.12.26 WriteLog ............................................................................. 243
1.4.12.27 WriteOutput ....................................................................... 243
1.4.12.28 WSGetAllChannels ............................................................... 243
1.4.12.29 WSGetSubscribers ................................................................. 243
1.4.12.30 WSPublish ......................................................................... 244
1.4.12.31 WSSendMessage ................................................................ 244
1.4.12.32 XmlChildPos ..................................................................... 244
1.4.12.33 XmlElemNew ..................................................................... 244
1.4.12.34 XmIFormat .......................................................................... 244
1.4.12.35 XmlGetNodeType ................................................................ 244
1.4.12.36 XmlNew ............................................................................... 245
1.4.12.37 XmlParse ........................................................................... 245
1.4.12.38 XmlSearch ......................................................................... 245
1.4.12.39 XmlTransform .................................................................... 246
1.4.12.40 XmlValidate ....................................................................... 246
1.4.12.41 Year ............................................................................... 246
1.4.12.42 YesNoFormat ...................................................................... 247

1.5 Ajax JavaScript Functions .................................................................. 247
  1.5.1 Function summary Ajax .................................................................. 247
  1.5.2 ColdFusion.Ajax.submitForm ..................................................... 247
  1.5.3 ColdFusion.Autosuggest.getAutosuggestObject ......................... 248
    1.5.3.1 ColdFusion.Chart.getChartHandle ...................................... 248
    1.5.3.2 ColdFusion.FileUpload.cancelUpload .................................. 248
    1.5.3.3 ColdFusion.FileUpload.clearAllFiles .................................. 248
    1.5.3.4 ColdFusion.fileUpload.getUrl ........................................... 248
    1.5.3.5 ColdFusion.FileUpload.startUpload .................................. 249
    1.5.3.6 ColdFusion.getElementById ................................................. 249
    1.5.3.7 ColdFusion.grid.clearSelectedRows .................................. 249
    1.5.3.8 ColdFusion.Grid.getBottomToolbar ................................... 249
    1.5.3.9 ColdFusion.Grid.getGridObject ......................................... 250
    1.5.3.10 ColdFusion.grid.getselectedRows .................................. 250
    1.5.3.11 ColdFusion.Grid.getTopToolbar ....................................... 250
    1.5.3.12 ColdFusion.Grid.hideBottomToolbar ................................ 250
    1.5.3.13 ColdFusion.Grid.hideTopToolbar ..................................... 250
    1.5.3.14 ColdFusion.Grid.refresh ................................................. 250
    1.5.3.15 ColdFusion.Grid.refreshBottomToolbar ........................... 250
    1.5.3.16 ColdFusion.Grid.refreshTopToolbar .................................. 251
    1.5.3.17 ColdFusion.Grid.showBottomToolbar ................................ 251
    1.5.3.18 ColdFusion.Grid.showTopToolbar ..................................... 251
    1.5.3.19 ColdFusion.Grid.sort ....................................................... 251
    1.5.3.20 ColdFusion.JSON.decode ............................................... 251
    1.5.3.21 ColdFusion.JSON.encode ................................................ 251
    1.5.3.22 ColdFusion.Layout.createTab ......................................... 251
    1.5.3.23 ColdFusion.Layout.collapseArea ...................................... 252
    1.5.3.24 ColdFusion.Layout.collapseAccordion .............................. 252
    1.5.3.25 ColdFusion.Layout.createAccordionPanel ........................ 252
    1.5.3.26 ColdFusion.Layout.enableSourceBind ................................ 252

  1.5.4 ColdFusion.Layout.enableSourceBind ........................................ 252
    1.5.4.1 ColdFusion.Layout.expandAccordion .................................. 252
    1.5.4.2 ColdFusion.Layout.disableTab ........................................... 253
    1.5.4.3 ColdFusion.Layout.enableTab ............................................ 253
    1.5.4.4 ColdFusion.Layout.expandArea ......................................... 253
    1.5.4.5 ColdFusion.Layout.getAccordionLayout ......................... 253
    1.5.4.6 ColdFusion.Layout.getBorderLayout ................................... 253
    1.5.4.7 ColdFusion.Layout.getTabLayout ...................................... 253
    1.5.4.8 ColdFusion.Layout.hideAccordion ................................... 254
    1.5.4.9 ColdFusion.Layout.hideArea .............................................. 254
1.5.4.10 ColdFusion.Layout.hideTab
1.5.4.11 ColdFusion.Layout.selectAccordion
1.5.4.12 ColdFusion.Layout.selectTab
1.5.4.13 ColdFusion.Layout.showAccordion
1.5.4.14 ColdFusion.Layout.showArea
1.5.4.15 ColdFusion.Layout.showTab
1.5.4.16 ColdFusion.Log.debug
1.5.4.17 ColdFusion.Log.dump
1.5.4.18 ColdFusion.Log.error
1.5.4.19 ColdFusion.Log.info
1.5.4.20 ColdFusion.Map.addEventListener
1.5.4.21 ColdFusion.Map.addMarker
1.5.4.22 ColdFusion.Map.getLatitudeLongitude
1.5.4.23 ColdFusion.Map.getMapObject
1.5.4.24 ColdFusion.Map.hide
1.5.4.25 ColdFusion.Map.refresh
1.5.4.26 ColdFusion.Map.setCenter
1.5.4.27 ColdFusion.Map.setZoomLevel
1.5.4.28 ColdFusion.Map.show
1.5.4.29 ColdFusion.MediaPlayer.getPlayer
1.5.4.30 ColdFusion.MediaPlayer.getType
1.5.4.31 ColdFusion.MediaPlayer.logError
1.5.4.32 ColdFusion.MediaPlayer.resize
1.5.4.33 ColdFusion.MediaPlayer.setTitle
1.5.4.34 ColdFusion.MediaPlayer.setMute
1.5.4.35 ColdFusion.MediaPlayer.getSource
1.5.4.36 ColdFusion.MediaPlayer.setVolume
1.5.4.37 ColdFusion.MediaPlayer.startPlay
1.5.4.38 ColdFusion.MediaPlayer.stopPlay
1.5.4.39 ColdFusion.MessageBox.create
1.5.4.40 ColdFusion.MessageBox.show
1.5.5 ColdFusion.MessageBox.getMessageBoxObject
1.5.6 ColdFusion.MessageBox.isMessageBoxDefined
1.5.6.1 ColdFusion.MessageBox.update
1.5.6.2 ColdFusion.MessageBox.updateMessage
1.5.6.3 ColdFusion.MessageBox.updateTitle
1.5.6.4 ColdFusion.navigate
1.5.7 ColdFusion.ProgressBar.getProgressBarObject
1.5.7.1 ColdFusion.ProgressBar.hide
1.5.7.2 ColdFusion.ProgressBar.reset
1.5.7.3 ColdFusion.ProgressBar.show
1.5.7.4 ColdFusion.ProgressBar.start
1.5.7.5 ColdFusion.ProgressBar.stop
1.5.7.6 ColdFusion.ProgressBar.update
1.5.7.7 ColdFusion.ProgressBar.updateStatus
1.5.7.8 ColdFusion.RichText.getDocumentObject
1.5.7.9 ColdFusion.RichText.onComplete
1.5.7.10 ColdFusion.setGlobalErrorHandler
1.5.7.11 ColdFusion.Slider.disable
1.5.7.12 ColdFusion.Slider.enable
1.5.7.13 ColdFusion.Slider.getSliderObject
1.5.7.14 ColdFusion.Slider.getValue
1.5.7.15 ColdFusion.Slider.hide
1.5.7.16 ColdFusion.Slider.setValue
1.5.7.17 ColdFusion.Slider.show
1.5.7.18 ColdFusion.Tree.getTreeObject
1.5.7.19 ColdFusion.Tree.refresh
1.5.7.20 ColdFusion.Window.create
1.5.7.21 ColdFusion.Window.destroy
1.5.7.22 ColdFusion.Window.getWindowObject
1.5.7.23 ColdFusion.Window.hide
1.5.7.24 ColdFusion.Window.onHide
1.5.7.25 ColdFusion.Window.onShow
1.5.7.26 ColdFusion.Window.show
1.5.8 JavaScriptFunctionsinColdFusion9Update1
1.6 Script Functions Implemented as CFCs
1.6.1 Accessing the functions
1.6.2 Function summary
1.6.3 ftp
1.6.4 htp
1.6.5 mail
1.6.6 pdf
1.6.7 query .......................................................... 269
1.6.8 storedproc .................................................. 270
1.6.9 Script functions implemented as CFCs in ColdFusion 9 Update 1 .................................................. 270
1.7 ColdFusion Flash Form Style Reference .......................................................... 272
  1.7.1 Styles valid for all controls .................................................. 272
  1.7.2 Styles for cfform .................................................. 272
  1.7.3 Styles for cfformgroup with horizontal or vertical type attributes .................................................. 272
  1.7.4 Styles for box-style cfformgroup elements .................................................. 272
  1.7.5 Styles for cfformgroup with accordion type attribute .................................................. 273
  1.7.6 Styles for cfformgroup with tabnavigator type attribute .................................................. 273
  1.7.7 Styles for cfformitem with hrule or vrule type attributes .................................................. 273
  1.7.8 Styles for cfformitem with radio, checkbox, button, image, or submit type attributes .................................................. 273
  1.7.9 Styles for cfformitem with text, password, or hidden type attributes .................................................. 273
  1.7.10 Styles for cfinput with size attribute value of 1 .................................................. 273
  1.7.11 Styles for cfinput with size attribute value greater than 1 .................................................. 273
  1.7.12 Styles for cfcalendar tag and cfinput with dateField type attribute .................................................. 273
  1.7.13 Styles for the cfgrid tag .................................................. 274
  1.7.14 Styles for the cffree tag .................................................. 274
1.8 Application.CFC Reference ............................................. 274
  1.8.1 Application variables .................................................. 274
  1.8.2 Method summary .................................................. 275
  1.8.3 onAbort .................................................. 275
  1.8.4 onApplicationEnd .................................................. 275
  1.8.5 onApplicationStart .................................................. 275
  1.8.6 onCFCRequest .................................................. 276
  1.8.7 onError .................................................. 276
  1.8.8 onMissingTemplate .................................................. 276
  1.8.9 onRequest .................................................. 276
  1.8.10 onRequestEnd .................................................. 276
  1.8.11 onRequestSessionStart .................................................. 277
  1.8.12 onRequestServerStart .................................................. 277
  1.8.13 onRequestSessionEnd .................................................. 277
  1.8.14 onRequestSessionStart .................................................. 277
1.9 ColdFusion Event Gateway Reference ............................................. 277
  1.9.1 addEvent .................................................. 277
  1.9.2 CFEvent .................................................. 278
  1.9.3 CFEventclass .................................................. 278
  1.9.4 Constructor .................................................. 278
  1.9.5 Gateway development interfaces and classes .................................................. 278
  1.9.6 GatewayHelper interface .................................................. 278
  1.9.7 Gateway interface .................................................. 278
  1.9.8 GatewayServices class .................................................. 278
  1.9.9 getCFCClass .................................................. 278
  1.9.10 getCFCTimeout .................................................. 279
  1.9.11 setCFCTimeout .................................................. 279
  1.9.12 setCFCTimeout .................................................. 279
  1.9.13 getData .................................................. 279
  1.9.14 getGatewayID .................................................. 279
  1.9.15 getGatewayID .................................................. 279
  1.9.16 setGatewayServices .................................................. 279
  1.9.17 getGatewayType .................................................. 279
  1.9.18 getHelper .................................................. 280
  1.9.19 getLogger .................................................. 280
  1.9.20 getMaxQueueSize .................................................. 280
  1.9.21 getOriginatorID .................................................. 280
  1.9.22 setCFCTimeout .................................................. 280
  1.9.23 setCFCTimeout .................................................. 280
  1.9.24 getQueueSize .................................................. 281
  1.9.25 getstatus .................................................. 281
  1.9.26 outgoingMessage .................................................. 281
  1.9.27 restart .................................................. 281
  1.9.28 setCFCListeners .................................................. 281
  1.9.29 setData .................................................. 281
  1.9.30 setGatewayID .................................................. 281
  1.9.31 setGatewayType .................................................. 282
  1.9.32 setOriginatorID .................................................. 282
  1.9.33 start .................................................. 282
  1.9.34 stop .................................................. 282
  1.9.35 USLogger class .................................................. 282
  1.9.36 debug .................................................. 282
  1.9.37 error .................................................. 283
  1.9.38 fatal .................................................. 283
Class libraries overview

- CCFXRequest class
- CCFXQuery class
- CCFXException class

SMS Gateway CFEvent structure and commands

- numberOfMessagesReceived
- getPermitList
- getDenyList
- getBuddyList
- addDeny
- addBuddy

IM gateway message sending commands

- IM Gateway GatewayHelper class methods
  - onBuddyStatus

IM gateway methods and commands

- submit command
- submit Multi command

- data command

1.9.79 CFML event gateway SendGatewayMessage data parameter

1.10 ColdFusion C++ CFX Reference

1.10.1 C++ class overview
1.10.2 Deprecated class methods
1.10.3 CCFXException class
1.10.4 CCFXQuery class
1.10.5 CCFXRequest class
1.10.6 CCFXStringSet class

1.11 ColdFusion Java CFX Reference

1.11.1 CFX libraries overview
1.11.2 Custom tag interface
1.11.3 Query interface
1.11.4 Request interface
1.11.5 Response interface
1.11.6 Debugging classes reference

1.12 WDDX JavaScript Objects

1.12.1 JavaScript object overview
1.12.2 WddxRecordset object
1.12.3 WddxSerializer object

1.13 ColdFusion ActionScript Functions

1.13.1 CF.http
1.13.2 CF.query

1.14 ColdFusion Mobile Functions

1.14.1 Accelerometer Functions
  1.14.1.1 cfclient.accelerometer.clearWatch
  1.14.1.2 cfclient.accelerometer.getOptions
  1.14.1.3 cfclient.accelerometer.setOptions
  1.14.1.4 cfclient.accelerometer.watch

1.14.2 Camera Functions
  1.14.2.1 cfclient.camera.cleanup
  1.14.2.2 cfclient.camera.getOptions
  1.14.2.3 cfclient.camera.getPicture
1.14.2.4 cfclient.camera.getPictureFromAlbum ........................................... 297
1.14.2.5 cfclient.camera.getPictureFromPhotoLibrary ................................ 298
1.14.2.6 cfclient.camera.setOptions .................................................. 298
1.14.2.7 options Object ...................................................................... 298
1.14.3 Connection Functions ............................................................... 298
  1.14.3.1 cfclient.connection.getType ............................................. 298
  1.14.3.2 cfclient.connection.onOffline ........................................ 298
  1.14.3.3 cfclient.connection.onOnline .......................................... 298
1.14.4 Contact Functions ...................................................................... 298
  1.14.4.1 Contact Creation Functions .............................................. 298
  1.14.4.2 Contacts Searching Functions ......................................... 299
1.14.5 Event Functions ......................................................................... 299
  1.14.5.1 cfclient.events.onBackButton ........................................ 299
  1.14.5.2 cfclient.events.onBatteryCritical ................................... 299
  1.14.5.3 cfclient.events.onBatteryLow ........................................ 299
  1.14.5.4 cfclient.events.onBatteryStatusChange ........................... 299
  1.14.5.5 cfclient.events.onMenuButton ....................................... 300
  1.14.5.6 cfclient.events.onPause ................................................ 300
  1.14.5.7 cfclient.events.onResume .............................................. 300
  1.14.5.8 cfclient.events.onSearchButton .................................... 300
1.14.6 File System Functions ................................................................ 300
  1.14.6.1 cfclient.file.append ..................................................... 300
  1.14.6.2 cfclient.file.copy .......................................................... 300
  1.14.6.3 cfclient.file.copyDirectory .......................................... 300
  1.14.6.4 cfclient.file.createDirectory ....................................... 300
  1.14.6.5 cfclient.file.directoryExists ....................................... 300
  1.14.6.6 cfclient.file.download .................................................. 301
  1.14.6.7 cfclient.file.exists ...................................................... 301
  1.14.6.8 cfclient.file.get ............................................................ 301
  1.14.6.9 cfclient.file.getDirectory ........................................... 301
  1.14.6.10 cfclient.file.getWorkingDirectory ................................ 301
  1.14.6.11 cfclient.file.listDirectory ......................................... 301
  1.14.6.12 cfclient.file.move ..................................................... 301
  1.14.6.13 cfclient.file.moveDirectory ....................................... 301
  1.14.6.14 cfclient.file.read ...................................................... 301
  1.14.6.15 cfclient.file.readAsBase64 ........................................ 301
  1.14.6.16 cfclient.file.rename .................................................. 302
  1.14.6.17 cfclient.file.renameDirectory .................................... 302
  1.14.6.18 cfclient.file.renameDirectory .................................... 302
  1.14.6.19 cfclient.file.setFileSystem ......................................... 302
  1.14.6.21 cfclient.file.upload .................................................. 302
  1.14.6.22 cfclient.file.write .................................................... 302
  1.14.6.23 DirectoryEntry Object .............................................. 302
  1.14.6.24 FileEntry Object ........................................................ 302
  1.14.6.25 rename Function .......................................................... 302
1.14.7 Geolocation Functions ............................................................ 303
  1.14.7.1 cfclient.geolocation.clearWatch .................................. 303
  1.14.7.2 cfclient.geolocation.getCurrentPosition ........................ 303
  1.14.7.3 cfclient.geolocation.getOptions .................................. 303
  1.14.7.4 cfclient.geolocation.setOptions .................................. 303
  1.14.7.5 cfclient.geolocation.watchPosition ................................. 303
1.14.8 Media and Capture Functions .................................................. 303
  1.14.8.1 Audio Functions .......................................................... 303
  1.14.8.2 Video Functions ........................................................... 304
1.14.9 Notification Functions ............................................................ 304
  1.14.9.1 cfclient.notification.alert .......................................... 304
  1.14.9.2 cfclient.notification.beep ............................................. 304
  1.14.9.3 cfclient.notification.confirm ..................................... 304
  1.14.9.4 cfclient.notification.vibrate ....................................... 304
1.14.10 Splash Screen Functions ....................................................... 304
  1.14.10.1 cfclient.splashscreen.hide ........................................ 304
  1.14.10.2 cfclient.splashscreen.show ....................................... 304
1.14.11 Storage Functions .................................................................... 304
  1.14.11.1 cfclient.localStorage.clear ....................................... 305
  1.14.11.2 cfclient.localStorage.getItem .................................... 305
  1.14.11.3 cfclient.localStorage.removeItem ................................ 305
  1.14.11.4 cfclient.localStorage.setItem .................................... 305
Adobe ColdFusion CFML Reference

The *CFML Reference* is your primary ColdFusion Markup Language (CFML) reference. Use this manual to learn about CFML tags and functions, ColdFusion expressions, and using JavaScript objects for WDDX in Adobe ColdFusion. It also provides detailed references for Java and C++ CFX interfaces.

- **CFML Reference (ColdFusion 11)**
- **Reserved Words and Variables**
- **ColdFusion Tags**
- **ColdFusion Functions**
- **Ajax JavaScript Functions**
- **Script Functions Implemented as CFCs**
- **ColdFusion Flash Form Style Reference**
- **Application.CFC Reference**
- **ColdFusion Event Gateway Reference**
- **ColdFusion C++ CFX Reference**
- **ColdFusion Java CFX Reference**
- **WDDX JavaScript Objects**
- **ColdFusion ActionScript Functions**
- **ColdFusion Mobile Functions**
New functions

The following new functions are added:

- `GetSafeHTML`
- `isSafeHTML`
- `ImageGetMetadata`
- `GeneratePBKDFKey`
- `SpreadSheetAddPagebreaks`
- `SpreadSheetAddAutofilter`
- `QueryExecute`
- `QueryGetRow`
- `ListMap`
- `ListReduce`
- `ListEach`
- `ArrayMap`
- `ArrayReduce`
- `StructMap`
- `StructReduce`
- `Serialize`
- `CanSerialize`
- `SerializeXML`
- `Deserialize`
- `CanDeSerialize`
- `DeserializeXML`
- `InvalidateOauthAccesstoken`
- `IsValidOauthAccesstoken`
- `EncodeForXpath`
- `EncodeForXMLAttribute`
- `InvokeCFClientFunction`

Changes in functions

The following functions are enhanced:

- `Canonicalize`
- `cflocation`
- `EncodeForCSS`
- `EncodeForHTML`
- `EncodeForHTMLAttribute`
- `EncodeForJavaScript`
- `EncodeForURL`
- `EncodeForXML`
- `SpreadsheetAddColumn`
- `SpreadsheetWrite`
- `SpreadsheetSetCellValue`
- `SpreadsheetAddRows`
- `SpreadsheetAddRow`
- `GetProfileSections`
- `SetProfileString`
- `GetProfileString`
New tags

The following tags have been added:

- cfimapfilter
- cfoauth
- cfclient
- cfclientsettings
- cfhtmltopdf
- cfhtmltopdfitem
- cf_socialplugin

Changes in tags

The following tags are enhanced:

- cfexchangecalendar
- cf_file_action = "uploadAll"
- cfmail
- cfzip
- cflogout
- clogin
- cfspreadsheet
- cfldap
- cflocation
- cfpdf
- cfwebsocket

Mobile functions

See ColdFusion Mobile Functions

Other changes

For a list of other changes, see New in ColdFusion.
Reserved Words and Variables

Adobe ColdFusion language includes reserved words and scope variables.

- Reserved words
- Scope-specific built-in variables
- Custom tag variables
- ColdFusion tag-specific variables
- CGI environment (CGI Scope) variables
Reserved words

The following list indicates words you must not use for ColdFusion variables, user-defined function names, or custom tag names. Although you can safely use some of these words in some situations, you can prevent errors by avoiding them entirely.

- Any name starting with cf. However, when you call a CFML custom tag directly, you prefix the custom tag page name with cf_
- Built-in function names, such as Now or Hash
- Scope names, such as Form or Session
- Operators, such as NE or IS
- The names of any built-in data structures, such as Error or File
- The names of any built-in variables, such as RecordCount or CGI variable names
- The following CFScript language element names:
  - for
  - default
  - continue
  - import
  - finally
  - local (inside function declaration)
  - interface
  - pageencoding

Remember that ColdFusion is not case sensitive. For example, all of the following are reserved words: IS, Is, iS, and is.

⚠️ Note

Keywords in the newly added statements, such as abort, rethrow, param, and so on are not reserved.
Reserved words in forms

Do not create form field names that end in any of the following, except to specify a form field validation rule by using a hidden form field name.

- _integer
- _float
- _range
- _date
- _time
- _eurodate
Reserved words in queries

The following table lists SQL keywords that are reserved in ColdFusion queries. This list includes all reserved words in the SQL standard. Avoid them in variables used in all queries. Do not use these keywords as variable names in any queries.

<table>
<thead>
<tr>
<th>ABSOLUTE</th>
<th>ACTION</th>
<th>ADD</th>
<th>ALL</th>
<th>ALLOCATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER</td>
<td>AND</td>
<td>ANY</td>
<td>ARE</td>
<td>AS</td>
</tr>
<tr>
<td>ASC</td>
<td>ASSERTION</td>
<td>AT</td>
<td>AUTHORIZATION</td>
<td>AVG</td>
</tr>
<tr>
<td>BEGIN</td>
<td>BETWEEN</td>
<td>BIT</td>
<td>BIT_LENGTH</td>
<td>BOTH</td>
</tr>
<tr>
<td>BY</td>
<td>CASCADE</td>
<td>CASCADED</td>
<td>CASE</td>
<td>CAST</td>
</tr>
<tr>
<td>CATALOG</td>
<td>CHAR</td>
<td>CHARACTER</td>
<td>CHARACTER_LENGTH</td>
<td>CHAR_LENGTH</td>
</tr>
<tr>
<td>CHECK</td>
<td>CLOSE</td>
<td>COALESCE</td>
<td>COLLATE</td>
<td>COLLATION</td>
</tr>
<tr>
<td>COLUMN</td>
<td>COMMIT</td>
<td>CONNECT</td>
<td>CONNECTION</td>
<td>CONSTRAINT</td>
</tr>
<tr>
<td>CONSTRAINTS</td>
<td>CONTINUE</td>
<td>CONVERT</td>
<td>CORRESPONDING</td>
<td>COUNT</td>
</tr>
<tr>
<td>CREATE</td>
<td>CROSS</td>
<td>CURRENT</td>
<td>CURRENT_DATE</td>
<td>CURRENT_TIME</td>
</tr>
<tr>
<td>CURRENT_TIMESTAMP</td>
<td>CURRENT_USER</td>
<td>CURSOR</td>
<td>DATE</td>
<td>DAY</td>
</tr>
<tr>
<td>DEALLOCATE</td>
<td>DEC</td>
<td>DECIMAL</td>
<td>DECLARE</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>DEFERRABLE</td>
<td>DEFERRED</td>
<td>DELETE</td>
<td>DESC</td>
<td>DESCRIBE</td>
</tr>
<tr>
<td>DESCRIPTOR</td>
<td>DIAGNOSTICS</td>
<td>DISCONNECT</td>
<td>DISTINCT</td>
<td>DOMAIN</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DROP</td>
<td>ELSE</td>
<td>END</td>
<td>END-EXEC</td>
</tr>
<tr>
<td>ESCAPE</td>
<td>EXCEPT</td>
<td>EXCEPTION</td>
<td>EXEC</td>
<td>EXECUTE</td>
</tr>
<tr>
<td>EXISTS</td>
<td>EXTERNAL</td>
<td>EXTRACT</td>
<td>FALSE</td>
<td>FETCH</td>
</tr>
<tr>
<td>FIRST</td>
<td>FLOAT</td>
<td>FOR</td>
<td>FOREIGN</td>
<td>FOUND</td>
</tr>
<tr>
<td>FROM</td>
<td>FULL</td>
<td>GET</td>
<td>GLOBAL</td>
<td>GO</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>GOTO</td>
<td>GRANT</td>
<td>GROUP</td>
<td>HAVING</td>
<td>HOUR</td>
</tr>
<tr>
<td>IDENTITY</td>
<td>IMMEDIATE</td>
<td>IN</td>
<td>INDICATOR</td>
<td>INITIALLY</td>
</tr>
<tr>
<td>INNER</td>
<td>INPUT</td>
<td>INSENSITIVE</td>
<td>INSERT</td>
<td>INT</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTERSECT</td>
<td>INTERVAL</td>
<td>INTO</td>
<td>IS</td>
</tr>
<tr>
<td>ISOLATION</td>
<td>JOIN</td>
<td>KEY</td>
<td>LANGUAGE</td>
<td>LAST</td>
</tr>
<tr>
<td>LEADING</td>
<td>LEFT</td>
<td>LEVEL</td>
<td>LIKE</td>
<td>LOCAL</td>
</tr>
<tr>
<td>LOWER</td>
<td>MATCH</td>
<td>MAX</td>
<td>MIN</td>
<td>MINUTE</td>
</tr>
<tr>
<td>MODULE</td>
<td>MONTH</td>
<td>NAMES</td>
<td>NATIONAL</td>
<td>NATURAL</td>
</tr>
<tr>
<td>NCHAR</td>
<td>NEXT</td>
<td>NO</td>
<td>NOT</td>
<td>NULL</td>
</tr>
<tr>
<td>NULLIF</td>
<td>NUMERIC</td>
<td>OCTET_LENGTH</td>
<td>OF</td>
<td>ON</td>
</tr>
<tr>
<td>ONLY</td>
<td>OPEN</td>
<td>OPTION</td>
<td>OR</td>
<td>ORDER</td>
</tr>
<tr>
<td>OUTER</td>
<td>OUTPUT</td>
<td>OVERLAPS</td>
<td>PAD</td>
<td>PARTIAL</td>
</tr>
<tr>
<td>POSITION</td>
<td>PRECISION</td>
<td>PREPARE</td>
<td>PRESERVE</td>
<td>PRIMARY</td>
</tr>
<tr>
<td>PRIOR</td>
<td>PRIVILEGES</td>
<td>PROCEDURE</td>
<td>PUBLIC</td>
<td>READ</td>
</tr>
<tr>
<td>REAL</td>
<td>REFERENCES</td>
<td>RELATIVE</td>
<td>RESTRICT</td>
<td>REVOKE</td>
</tr>
<tr>
<td>RIGHT</td>
<td>ROLLBACK</td>
<td>ROWS</td>
<td>SCHEMA</td>
<td>SCROLL</td>
</tr>
<tr>
<td>SECOND</td>
<td>SECTION</td>
<td>SELECT</td>
<td>SESSION</td>
<td>SESSION_USER</td>
</tr>
<tr>
<td>SET</td>
<td>SIZE</td>
<td>SMALLINT</td>
<td>SOME</td>
<td>SPACE</td>
</tr>
<tr>
<td>SQL</td>
<td>SQLCODE</td>
<td>SQLERROR</td>
<td>SQLSTATE</td>
<td>SUBSTRING</td>
</tr>
<tr>
<td>SUM</td>
<td>SYSTEM_USER</td>
<td>TABLE</td>
<td>TEMPORARY</td>
<td>THEN</td>
</tr>
<tr>
<td>TIME</td>
<td>TIMESTAMP</td>
<td>TIMEZONE_HOUR</td>
<td>TIMEZONE_MINUTE</td>
<td>TO</td>
</tr>
<tr>
<td>TRAILING</td>
<td>TRANSACTION</td>
<td>TRANSLATE</td>
<td>TRANSLATION</td>
<td>TRIM</td>
</tr>
<tr>
<td>TRUE</td>
<td>UNION</td>
<td>UNIQUE</td>
<td>UNKNOWN</td>
<td>UPDATE</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>UPPER</td>
<td>USAGE</td>
<td>USER</td>
<td>USING</td>
<td>VALUE</td>
</tr>
<tr>
<td>VALUES</td>
<td>VARCHAR</td>
<td>VARYING</td>
<td>VIEW</td>
<td>WHEN</td>
</tr>
<tr>
<td>WHenever</td>
<td>WHERE</td>
<td>WITH</td>
<td>WORK</td>
<td>WRITE</td>
</tr>
<tr>
<td>YEAR</td>
<td>ZONE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scope-specific built-in variables

ColdFusion returns variables, such as those returned in a `cfdirectory` or `cfftp` operation. A variable is usually referenced by `scoping` it according to its type: naming it according to the code context in which it is available; for example, `Session.varname`, or `Application.varname`. For more information on ColdFusion scopes, see Using ColdFusion Variables in the Developing ColdFusion Applications

You use the `cflock` tag to limit the scope of CFML constructs that modify shared data structures, files, and CFXs, to ensure that modifications occur sequentially. For more information, see `cflock`, and Using Persistent Data and Locking in the Developing ColdFusion Applications.
Variable scope

ColdFusion supports the Variables scope. Unscoped variables created with the \texttt{cfset} tag acquire the Variables scope by default. For example, the variable created by the statement \texttt{<CFSET linguist = Chomsky>} can be referenced as{{#Variables.linguist#}}.
Caller scope

History

ColdFusion MX: The Caller scope is accessible as a structure. (In earlier releases, it was not.)
CGI variables

see CGI environment (CGI Scope) variables
Client variables

The following client variables are reserved:

- `Client.CFID`
- `Client.CFToken`
- `Client.HitCount`
- `Client.LastVisit`
- `Client.TimeCreated`
- `Client.URLToken`
Server variables

Use the Server prefix to reference server variables, as follows:

- Server.ColdFusion.ProductName
- Server.ColdFusion.ProductVersion
- Server.ColdFusion.ProductLevel
- Server.ColdFusion.SerialNumber
- Server.ColdFusion.ShippedLocales
- Server.ColdFusion.AppServer
- Server.ColdFusion.Expiration
- Server.ColdFusion.RootDir
- Server.ColdFusion.UpdateLevel
- Server.OS.Name
- Server.OS.AdditionalInformation
- Server.OS.Version
- Server.OS.BuildNumber
### Application and session variables

To enable application and session variables, use the `cfapplication` tag or `Application.cfc`. Reference them as follows:

<table>
<thead>
<tr>
<th>Application</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>myvariable</td>
<td>myvariable</td>
</tr>
</tbody>
</table>

To ensure that modifications to shared data occur in the intended sequence, use the `cflock` tag. For more information, see [cflock](#).

ColdFusion provides the following predefined application and session variables:

<table>
<thead>
<tr>
<th>Application</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationName</td>
<td>CFID</td>
</tr>
<tr>
<td></td>
<td>CFToken</td>
</tr>
<tr>
<td></td>
<td>URLToken</td>
</tr>
</tbody>
</table>
Custom tag variables

A ColdFusion custom tag returns the following variables:

- `ThisTag.ExecutionMode`
- `ThisTag.HasEndTag`
- `ThisTag.GeneratedContent`
- `ThisTag.AssocAttribs[index]`

A custom tag can set a Caller variable to provide information to the caller. Set the Caller variable as follows:

```cfset Caller.variable_name = "value"```

The calling page can access the variable with the `cfoutput` tag, as follows:

```<cfoutput>#variable_name#</cfoutput>```
Request variable

Request variables store data about the processing of one page request. Request variables store data in a structure that can be passed to nested tags, such as custom tags, and processed once. To provide information to nested tags, set a Request variable, as follows:

```
<CFSET Request.field_name1 = "value">
<CFSET Request.field_name2 = "value">
<CFSET Request.field_name3 = "value">
...
```

Each nested tag can access the variable with the `cfoutput` tag, as follows:

```
<CFOUTPUT>#Request.field_name1#</CFOUTPUT>
```
Form variable

ColdFusion supports the Form variable FieldNames. FieldNames returns the names of the fields on a form. You use it on the action page associated with a form, as follows:

```
Form.FieldNames
```

#back to top
ColdFusion tag-specific variables

Some ColdFusion tags return data as variables. For example, the `cfile` tag returns file size information in the FileSize variable, referenced as CFFILE.FileSize.

The following tags return data that you can reference in variables:

catch
cdirectory
cerror
cfile
cftp
chttp
cindex
cldap
cpop
cquery
cregistry
csearch
cstoredproc
ColdFusion query variables

A ColdFusion tag that returns a query object supports the following variables, where *queryname* is the value of the `name` attribute:

```plaintext
queryname.CurrentRow
queryname.RecordCount
queryname.ColumnList
```
CFCATCH variables

Within a cfcatch block, the active exception properties can be accessed as the following variables:

```
CFCATCH.Type
CFCATCH.Message
CFCATCH.Detail
CFCATCH.ErrNumber
CFCATCH.NativeErrorCode
CFCATCH.SQLState
CFCATCH.LockName
CFCATCH.LockOperation
CFCATCH.MissingFileName
CFCATCH.TagContext
CFCATCH.ErrorCode
CFCATCH.ExtendedInfo
```

Within a cfcatch block, database exception properties can be accessed as the following variables:

```
CFCATCH.QueryError
CFCATCH.SQL
CFCATCH.Where
CFCATCH.Datasource
```

Within a cfcatch block, undefined variable exception properties can be accessed as the following variable:

```
CFCATCH.Name
```

Within a cfcatch block, syntax and parsing exception properties can be accessed as the following variables:

```
CFCATCH.TokenText
CFCATCH.Snippet
CFCATCH.Column
CFCATCH.KnownColumn
CFCATCH.Line
CFCATCH.KnownLine
```
CFDIRECTORY variables

The `cfdirectory` tag, with `action=list`, returns a query object as follows, where `queryname` is the `name` attribute value:

```
queryname.Name
queryname.Size
queryname.Type
queryname.DateLastModified
queryname.Attributes
queryname.Mode
```
**CFERROR variables**

When `cferror` generates an error page, the following error variables are available if `type=`"request" or "exception".

- `Error.Diagnostics`
- `Error.MailTo`
- `Error.DateTime`
- `Error.Browser`
- `Error.GeneratedContent`
- `Error.RemoteAddress`
- `Error.HTTPReferer`
- `Error.Template`
- `Error.QueryString`

The following error variables are available if `type=`"validation".

- `Error.ValidationHeader`
- `Error.InvalidFields`
- `Error.ValidationFooter`

Any `cfcatch` variable that applies to exception type can be accessed within the `Error` scope, as follows:

- `Error.Type`
- `Error.Message`
- `Error.Detail`
- `Error.ErrNumber`
- `Error.NativeErrorCode`
- `Error.SQLState`
- `Error.LockName`
- `Error.LockOperation`
- `Error.MissingFileName`
- `Error.TagContext`
- `Error.ErrorCode`
- `Error.ExtendedInfo`

⚠️ **Note**

You can substitute the prefix `CFERROR` for `Error`, if `type = "Exception"`; for example, `CFERROR.Diagnostics`, `CFERROR.Mailto`, or `CFERROR.DateTime`. 
CFFILE ACTION=Upload variables

File variables are read-only. Use the CFFILE prefix to reference file variables, for example, CFFILE.ClientDirectory. The File prefix is deprecated in favor of the CFFILE prefix.

CFFILE.AttemptedServerFile
CFFILE.ClientDirectory
CFFILE.ClientFile
CFFILE.ClientFileExt
CFFILE.ClientFileName
CFFILE.ContentSubType
CFFILE.ContentType
CFFILE.DateLastAccessed
CFFILE.FileExisted
CFFILE.FileSize
CFFILE.FileWasAppended
CFFILE.FileWasOverwritten
CFFILE.FileWasRenamed
CFFILE.FileWasSaved
CFFILE.OldFileSize
CFFILE.ServerDirectory
CFFILE.ServerFile
CFFILE.ServerFileExt
CFFILE.ServerFileName
CFFILE.TimeCreated
CFFILE.TimeLastModified
CFFTP error variables

When you use the cfftp stoponerror attribute, the following variables are populated:

- CFFTP.Succeeded
- CFFTP.ErrorCode
- CFFTP>ErrorText
CFFTP ReturnValue variable

Some cfftp file and directory operations provide a return value, in the variable CFFTP.ReturnValue. Its value is determined by the results of the action attribute. When you specify any of the following actions, cfftp returns a value:

- GetCurrentDir
- GetCurrentURL
- ExistsDir
- ExistsFile
- Exists
### CFFTP query object columns

When you use the `cfftp` tag with the `listdir` action, `cfftp` returns a query object, where `queryname` is the `name` attribute value, and `row__` is the row number of each file or directory entry:

| `queryname.Name[row]`   |
| `queryname.Path[row]`   |
| `queryname.URL[row]`    |
| `queryname.Length[row]` |
| `queryname.LastModified[row]` |
| `queryname.Attributes` |
| `queryname.isDirectory` |
| `queryname.Mode`        |
CFHTTP variables

A `cfhttp` get operation can return text and binary files. Files are downloaded and the contents stored in a variable or file, depending on the MIME type, as follows:

- `CFHTTP.FileContent`
- `CFHTTP.MimeType`
- `CFHTTP.Header`
- `CFHTTP.ResponseHeader[http_hd_key]`
- `CFHTTP.StatusCode`
CFLDAP variables

The `cfldap` action=query tag returns information about the LDAP query, as follows:

queryname.CurrentRow
queryname.RecordCount
queryname.ColumnList
CFPOP variables

The `cfpop` tag returns the following result columns, depending on the `action` attribute value and the use of other attributes, such as `attachmentpath`, where `queryname` is the `name` attribute value:

<table>
<thead>
<tr>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>queryname.Date</code></td>
</tr>
<tr>
<td><code>queryname.From</code></td>
</tr>
<tr>
<td><code>queryname.Body</code></td>
</tr>
<tr>
<td><code>queryname.Header</code></td>
</tr>
<tr>
<td><code>queryname.MessageNumber</code></td>
</tr>
<tr>
<td><code>queryname.ReplyTo</code></td>
</tr>
<tr>
<td><code>queryname.Subject</code></td>
</tr>
<tr>
<td><code>queryname.CC</code></td>
</tr>
<tr>
<td><code>queryname.To</code></td>
</tr>
<tr>
<td><code>queryname.CurrentRow</code></td>
</tr>
<tr>
<td><code>queryname.RecordCount</code></td>
</tr>
<tr>
<td><code>queryname.ColumnHeader</code></td>
</tr>
<tr>
<td><code>queryname.Attachments</code></td>
</tr>
<tr>
<td><code>queryname.AttachmentFiles</code></td>
</tr>
</tbody>
</table>
CFQUERY and CFSTOREDPROC variables

The `<cfquery>` tag returns information about the query in this variable:

```
CFQUERY.ExecutionTime
```

The `<cfquery>` tag uses the query name to scope the following data about the query:

```
queryname.CurrentRow
queryname.RecordCount
queryname.ColumnList
```

The `<cfstoredproc>` tag returns the following variables:

```
CFSTOREDPROC.ExecutionTime
CFSTOREDPROC.StatusCode
```
CFREGISTRY variables

The cfregistry tag returns a query record set that you can reference after executing the GetAll action, as follows, where *queryname* is the name attribute value:

```plaintext
queryname.Entry
queryname.Type
queryname.Value
```
CFSEARCH variables

A cfsearch operation returns the following variables, where searchname is the name attribute value:

- searchname.URL
- searchname.Key
- searchname.Title
- searchname.Score
- searchname.Custom1 and Custom2
- searchname.Summary
- searchname.RecordCount
- searchname.CurrentRow
- searchname.RecordsSearched
- searchname.ColumnList
CGI environment (CGI Scope) variables

When a browser makes a request to a server, the web server and the browser create environment variables. In ColdFusion, these variables are referred to as *CGI environment variables*. CGI Environment variables contain data about the transaction between the browser and the server, such as the IP Address, browser type, and authenticated username. The available CGI variables depend on the browser and server software.

The CGI variables are available to ColdFusion pages in the CGI scope. They take the CGI prefix regardless of whether the server uses a server API or CGI to communicate with the ColdFusion server. You can reference CGI environment variables for a given page request anywhere in the page. CGI variables are read-only.

By default, when you use the `cfdump` tag to display the CGI scope, or when you request debug output of the CGI scope, ColdFusion attempts to display a fixed list of standard CGI environment variables. Because the available variables depend on the server, browser, and the types of interactions between the two, not all variables are normally available. They are represented by empty strings in the debug output. You can request any CGI variable in your application code, including variables that are not in the list variables displayed by dump and debug output. ColdFusion checks for the following variables for the `cfdump` tag and debug output:
The following sections describe how to test for CGI environment variables and provide information on some of the more commonly used CGI environment variables.

AUTH_PASSWORD
AUTH_TYPE
AUTH_USER
CERT_COOKIE
CERT_FLAGS
CERT_ISSUER
CERT_KEYSIZE
CERT_SECRETKEYSIZE
CERT_SERIALNUMBER
CERT_SERVER_ISSUER
CERT_SERVER_SUBJECT
CERT_SUBJECT
CF_TEMPLATE_PATH
CONTENT_LENGTH
CONTENT_TYPE
CONTEXT_PATH
GATEWAY_INTERFACE
HTTPS
HTTPS_KEYSIZE
HTTPS_SECRETKEYSIZE
HTTPS_SERVER_ISSUER
HTTPS_SERVER_SUBJECT
HTTP_ACCEPT
HTTP_ACCEPT_ENCODING
HTTP_ACCEPT_LANGUAGE
HTTP_CONNECTION
HTTP_COOKIE
HTTP_HOST
HTTP_REFERER
HTTP_USER_AGENT
QUERY_STRING
REMOTE_ADDR
REMOTE_HOST
REMOTE_USER
REQUEST_METHOD
SCRIPT_NAME
SERVER_NAME
SERVER_PORT
SERVER_PORT_SECURE
SERVER_PROTOCOL
SERVER_SOFTWARE
WEB_SERVER_API (This value is always blank; retained for compatibility.)
Testing for CGI variables

Because some browsers do not support some CGI variables, ColdFusion always returns true when it tests for the existence of a CGI variable, regardless of whether the browser supports the variable. To determine if the CGI variable is available, test for an empty string, as the following example shows:

```cfml
<cfif CGI.varname IS NOT "">
  CGI variable exists
<cfelse>
  CGI variable does not exist
</cfif>
```
# CGI server variables

The following table describes common CGI environment variables that the server creates (some variables are not available with some servers):

<table>
<thead>
<tr>
<th>CGI server variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVER_SOFTWARE</td>
<td>Name and version of the information server software answering the request (and running the gateway). Format: name/version.</td>
</tr>
<tr>
<td>SERVER_NAME</td>
<td>Server’s hostname, DNS alias, or IP address as it appears in self-referencing URLs.</td>
</tr>
<tr>
<td>GATEWAY_INTERFACE</td>
<td>CGI specification revision with which this server complies. Format: CGI/revision.</td>
</tr>
<tr>
<td>SERVER_PROTOCOL</td>
<td>Name and revision of the information protocol this request came in with. Format: protocol/revision.</td>
</tr>
<tr>
<td>SERVER_PORT</td>
<td>Port number to which the request was sent.</td>
</tr>
<tr>
<td>REQUEST_METHOD</td>
<td>Method with which the request was made. For HTTP, this is Get, Head, Post, and so on.</td>
</tr>
<tr>
<td>PATH_INFO</td>
<td>Extra path information, as given by the client. Scripts can be accessed by their virtual path name, followed by extra information at the end of this path. The extra information is sent as PATH_INFO.</td>
</tr>
<tr>
<td>PATH_TRANSLATED</td>
<td>Translated version of PATH_INFO after any virtual-to-physical mapping.</td>
</tr>
<tr>
<td>SCRIPT_NAME</td>
<td>Virtual path to the script that is executing; used for self-referencing URLs.</td>
</tr>
<tr>
<td>QUERY_STRING</td>
<td>Query information that follows the ? in the URL that referenced this script.</td>
</tr>
<tr>
<td>REMOTE_HOST</td>
<td>Hostname making the request. If the server does not have this information, it sets REMOTE_ADDR and does not set REMOTE_HOST.</td>
</tr>
<tr>
<td>REMOTE_ADDR</td>
<td>IP address of the remote host making the request.</td>
</tr>
<tr>
<td>AUTH_TYPE</td>
<td>If the server supports user authentication, and the script is protected, the protocol-specific authentication method used to validate the user.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REMOTE_USER</td>
<td>If the server supports user authentication, and the script is protected, the username the user has authenticated as. (Also available as AUTH_USER.)</td>
</tr>
<tr>
<td>REMOTE_IDENT</td>
<td>If the HTTP server supports RFC 931 identification, this variable is set to the remote username retrieved from the server. Use this variable for logging only.</td>
</tr>
<tr>
<td>CONTENT_TYPE</td>
<td>For queries that have attached information, such as HTTP POST and PUT, this is the content type of the data.</td>
</tr>
<tr>
<td>CONTENT_LENGTH</td>
<td>Length of the content as given by the client.</td>
</tr>
</tbody>
</table>
## CGI client variables

The following table describes common CGI environment variables the browser creates and passes in the request header:

<table>
<thead>
<tr>
<th>CGI client variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP_REFERER</td>
<td>The referring document that linked to or submitted form data.</td>
</tr>
<tr>
<td>HTTP_USER_AGENT</td>
<td>The browser that the client is currently using to send the request. Format: software/version library/version.</td>
</tr>
<tr>
<td>HTTP_IF_MODIFIED_SINCE</td>
<td>The last time the page was modified. The browser determines whether to set this variable, usually in response to the server having sent the <code>LAST_MODIFIED</code> HTTP header. It can be used to take advantage of browser-side caching.</td>
</tr>
</tbody>
</table>
ColdFusion makes available the following client certificate data. These variables are available when running Microsoft IIS 4.0 or Netscape Enterprise under SSL if your web server is configured to accept client certificates.

<table>
<thead>
<tr>
<th>CGI client certificate variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERT_SUBJECT</td>
<td>Client-specific information provided by the web server. This data typically includes the client's name, e-mail address, and so on, for example: O = &quot;VeriSign, Inc.&quot;, OU = VeriSign Trust Network, OU = &quot;www.verisign.com/repository/RPA Incorp. by Ref.,LIAB.LTD(c)98&quot;, OU = Persona Not Validated, OU = Digital ID Class 1 - Microsoft, CN = Matthew Lund, E = <a href="mailto:mlund@.com">mlund@.com</a></td>
</tr>
<tr>
<td>CERT_ISSUER</td>
<td>Information about the authority that provided the client certificate, for example: O = &quot;VeriSign, Inc.&quot;, OU = VeriSign Trust Network, OU = &quot;www.verisign.com/repository/RPA Incorp. by Ref.,LIAB.LTD(c)98&quot;, CN = VeriSign Class 1 CA Individual Subscriber-Persona Not Validated</td>
</tr>
</tbody>
</table>
ColdFusion Tags

ColdFusion Markup Language (CFML) includes a set of tags that you use in ColdFusion pages to interact with data sources, manipulate data, and display output. CFML tag syntax is similar to HTML element syntax.

- Tags in ColdFusion 10
- Tag summary
- Tags by function
- Tag changes since ColdFusion 5
- Tags a-b
- Tags c
- Tags d-e
- Tags f
- Tags g-h
- Tags i
- Tags j-l
- Tags m-o
- Tags p-q
- Tags r-s
- Tags t
- Tags u-z

#back to top
Tags in ColdFusion 10

The following table briefly describes CFML tags added in ColdFusion 10:

<table>
<thead>
<tr>
<th>CFML tag</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfexchangeconversation</td>
<td>Communications tags</td>
<td>Helps users organize and manage conversations from a Microsoft Exchange account.</td>
</tr>
<tr>
<td>cfexchangefolder</td>
<td>Communications tags</td>
<td>Allows you to perform various actions on the mail folder, such as get folder information, find folders, or create, copy, modify, move, delete, and empty the contents of a folder.</td>
</tr>
<tr>
<td>cfwebsocket</td>
<td>Web Socket tags</td>
<td>Lets you create the WebSocket object in your CFM template. The tag creates a reference to the WebSocket JavaScript object at the client-side.</td>
</tr>
</tbody>
</table>
### Tag summary

The following table briefly describes CFML tags:

<table>
<thead>
<tr>
<th>CFML tag</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfabort</td>
<td>Flow-control tags</td>
<td>Stops the processing of a ColdFusion page at the tag location</td>
</tr>
<tr>
<td>cfajaximport</td>
<td>Internet protocol tags</td>
<td>Controls importation of JavaScript files used for ColdFusion AJAX-based features</td>
</tr>
<tr>
<td>cfajaxproxy</td>
<td>Internet protocol tags</td>
<td>Generates an AJAX proxy class on the client page for a ColdFusion component</td>
</tr>
<tr>
<td>cfapplet</td>
<td>Forms tags</td>
<td>Embeds Java applets in a cfform tag</td>
</tr>
<tr>
<td>cfapplication</td>
<td>Application framework tags</td>
<td>Defines an application name; activates client variables; specifies client variable storage mechanism</td>
</tr>
<tr>
<td>cfargument</td>
<td>Extensibility tags</td>
<td>Creates a parameter definition within a component definition; defines a function argument</td>
</tr>
<tr>
<td>cfassociate</td>
<td>Application framework tags</td>
<td>Enables subtag data to be saved with a base tag</td>
</tr>
<tr>
<td>cfbreak</td>
<td>Flow-control tags</td>
<td>Breaks out of a CFML looping construct</td>
</tr>
<tr>
<td>cfcache</td>
<td>Page processing tags</td>
<td>Caches ColdFusion pages</td>
</tr>
<tr>
<td>cfcalendar</td>
<td>Forms tags</td>
<td>Provides a calendar from which to select a date</td>
</tr>
<tr>
<td>cfcase</td>
<td>Flow-control tags</td>
<td>Used with the cfswitch and cfdefaultcase tags</td>
</tr>
<tr>
<td>cfcatch</td>
<td>Exception handling tags, Flow-control tags</td>
<td>Catches exceptions in ColdFusion pages</td>
</tr>
<tr>
<td>cfchart</td>
<td>Data output tags</td>
<td>Generates and displays a chart</td>
</tr>
<tr>
<td>cfchartdata</td>
<td>Data output tags</td>
<td>Defines chart data points</td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cfchartseries</td>
<td>Data output tags</td>
<td>Defines style in which chart data displays</td>
</tr>
<tr>
<td>cfclient</td>
<td></td>
<td>The <code>&lt;cfClient&gt;</code> tag is a marker tag that instructs ColdFusion to generate client-side code (JavaScript) for ColdFusion code.</td>
</tr>
<tr>
<td>cfclientsettings</td>
<td></td>
<td>This tag is similar to <code>cfprocessingdirective</code> and acts as a compiler directive to include plugins for various features (device detection and device API).</td>
</tr>
<tr>
<td>cfcol</td>
<td>Data output tags</td>
<td>Defines table column header, properties</td>
</tr>
<tr>
<td>cfcollection</td>
<td>Extensibility tags</td>
<td>Administers Solr collections</td>
</tr>
<tr>
<td>cfcomponent</td>
<td>Extensibility tags</td>
<td>Creates and defines a component object</td>
</tr>
<tr>
<td>cfcontent</td>
<td>Data output tags, Page processing tags</td>
<td>Defines content type and filename of a file to be downloaded by the current page</td>
</tr>
<tr>
<td>cfcontinue</td>
<td>Flow-control tags</td>
<td>Returns processing to the top of a loop; used within a <code>cfloop</code> tag.</td>
</tr>
<tr>
<td>cfcookie</td>
<td>Variable manipulation tags</td>
<td>Defines and sets cookie variables, including expiration and security options</td>
</tr>
<tr>
<td>cfdbinfo</td>
<td>Database manipulation tags</td>
<td>Lets you retrieve information about a data source</td>
</tr>
<tr>
<td>cfdefaultcase</td>
<td>Flow-control tags</td>
<td>Receives control if there is no matching <code>cfcase</code> tag value</td>
</tr>
<tr>
<td>cfdirectory</td>
<td>File management tags</td>
<td>Performs typical directory-handling tasks from within a ColdFusion application</td>
</tr>
<tr>
<td>cfdiv</td>
<td>Display management tags</td>
<td>Creates an HTML tag with that is populated using a bind expressions.</td>
</tr>
<tr>
<td>cfdocument</td>
<td>Data output tags</td>
<td>Creates PDF or Adobe FlashPaper output from a text block that contains CFML and HTML</td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cfdocumentitem</td>
<td>Data output tags</td>
<td>Specifies action items, such as header, footer, and page break, for a PDF or FlashPaper document</td>
</tr>
<tr>
<td>cfdocumentsection</td>
<td>Data output tags</td>
<td>Divides a PDF or FlashPaper document into sections</td>
</tr>
<tr>
<td>cfdump</td>
<td>Debugging tags, Variable manipulation tags</td>
<td>Outputs variables for debugging</td>
</tr>
<tr>
<td>cfelse</td>
<td>Flow-control tags</td>
<td>Creates IF-THEN-ELSE constructs</td>
</tr>
<tr>
<td>cfelseif</td>
<td>Flow-control tags</td>
<td>Creates IF-THEN-ELSE constructs</td>
</tr>
<tr>
<td>cferror</td>
<td>Exception handling tags, Application framework tags</td>
<td>Displays custom HTML error pages when errors occur</td>
</tr>
<tr>
<td>cfexchangecalendar</td>
<td>Communications tags</td>
<td>Gets, creates, deletes, modifies, or responds to Microsoft Exchange calendar events</td>
</tr>
<tr>
<td>cfexchangeconnection</td>
<td>Communications tags</td>
<td>Opens or closes a persistent connection with an Exchange server</td>
</tr>
<tr>
<td>cfexchangecontact</td>
<td>Communications tags</td>
<td>Gets, creates, deletes, or modifies Exchange contacts</td>
</tr>
<tr>
<td>cfexchangeconversation</td>
<td>Communications tags</td>
<td>Helps users organize and manage conversations from a Microsoft Exchange account.</td>
</tr>
<tr>
<td>cfexchangefolder</td>
<td>Communications tags</td>
<td>Allows you to perform various actions on the mail folder, such as get folder information, find folders, create, modify, move, delete, and empty the contents of a folder.</td>
</tr>
<tr>
<td>cfexchangefilter</td>
<td>Communications tags</td>
<td>Sets filter conditions used in Exchange tag get operations</td>
</tr>
<tr>
<td>cfexchangemail</td>
<td>Communications tags</td>
<td>Gets and deletes Exchange mail messages and sets message properties</td>
</tr>
<tr>
<td>cfexchangetask</td>
<td>Communications tags</td>
<td>Gets, creates, deletes, or modifies an Exchange user task</td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cfexecute</td>
<td>Flow-control tags, Extensibility tags</td>
<td>Executes developer-specified process on server computer</td>
</tr>
<tr>
<td>cfexit</td>
<td>Flow-control tags</td>
<td>Aborts processing of executing CFML tag</td>
</tr>
<tr>
<td>cffeed</td>
<td>Communications tags, Internet protocol tags</td>
<td>Reads, creates, and converts, Atom and RSS syndication feeds</td>
</tr>
<tr>
<td>cffile</td>
<td>File management tags</td>
<td>Performs typical file-handling tasks from within ColdFusion application</td>
</tr>
<tr>
<td>cffileupload</td>
<td>File management tags, Forms tags</td>
<td>Displays a dialog for uploading multiple files from the user's system.</td>
</tr>
<tr>
<td>cffinally</td>
<td>Exception handling tags</td>
<td>Used inside a cftry tag</td>
</tr>
<tr>
<td>cfflush</td>
<td>Data output tags, Page processing tags</td>
<td>Flushes currently available data to client</td>
</tr>
<tr>
<td>cfform</td>
<td>Forms tags</td>
<td>Builds input form; performs client-side input validation</td>
</tr>
<tr>
<td>cfformgroup</td>
<td>Forms tags</td>
<td>Groups form control into a containing object</td>
</tr>
<tr>
<td>cfformitem</td>
<td>Forms tags</td>
<td>Adds text and dividing rules to Adobe Flash forms</td>
</tr>
<tr>
<td>cfftp</td>
<td>Forms tags, Extensibility tags, Internet protocol tags</td>
<td>Permits FTP file operations</td>
</tr>
<tr>
<td>cffunction</td>
<td>Extensibility tags</td>
<td>Defines function that you build in CFML</td>
</tr>
<tr>
<td>cfgrid</td>
<td>Forms tags</td>
<td>Displays tabular grid control, in cfgrid tag</td>
</tr>
<tr>
<td>cfgridcolumn</td>
<td>Forms tags</td>
<td>Used in cfform; defines columns in a cfgrid</td>
</tr>
<tr>
<td>cfgridrow</td>
<td>Forms tags</td>
<td>Defines a grid row; used with cfgrid</td>
</tr>
<tr>
<td>cfgridupdate</td>
<td>Forms tags</td>
<td>Directly updates ODBC data source from edited grid data</td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cfheader</td>
<td>Data output tags, Page processing tags</td>
<td>Generates HTTP headers</td>
</tr>
<tr>
<td>cfhtmlhead</td>
<td>Page processing tags</td>
<td>Writes text and HTML to HEAD section of page</td>
</tr>
<tr>
<td>cfhtmltopdf</td>
<td>Data output tags</td>
<td>&lt;cfhtmltopdf&gt; creates high quality PDF output from a text block containing CFML and HTML using the PDF Service Manager.</td>
</tr>
<tr>
<td>cfhtmltopdfitem</td>
<td>Data output tags</td>
<td>The &lt;cfhtmltopdfitem&gt; specifies the action items for a PDF document created by the &lt;cfhtmltopdf&gt;tag.</td>
</tr>
<tr>
<td>cfhttp</td>
<td>Internet protocol tags</td>
<td>Performs GET and POST to upload file or post form, cookie, query, or CGI variable directly to server</td>
</tr>
<tr>
<td>cfhttpparam</td>
<td>Internet protocol tags</td>
<td>Specifies parameters required for a cfhttp POST operation; used with cfhttp</td>
</tr>
<tr>
<td>cfif</td>
<td>Flow-control tags</td>
<td>Creates IF-THEN-ELSE constructs</td>
</tr>
<tr>
<td>cfimage</td>
<td>Other tags</td>
<td>Creates a cfimage, a ColdFusion data type that can be operated by image functions.</td>
</tr>
<tr>
<td>cfimap</td>
<td>Communications tags, Internet protocol tags</td>
<td>Retrieves and manages e-mails and folders in IMAP servers</td>
</tr>
<tr>
<td>cfimapfilter</td>
<td>Communication tags</td>
<td>Specifies filter parameters that control the actions of cfimap, get operations.</td>
</tr>
<tr>
<td>cfimport</td>
<td>Application framework tags</td>
<td>Imports JSP tag libraries into a CFML page</td>
</tr>
<tr>
<td>cfinclude</td>
<td>Flow-control tags</td>
<td>Embeds references to ColdFusion pages</td>
</tr>
<tr>
<td>cfindex</td>
<td>Extensibility tags</td>
<td>Creates Solr search indexes</td>
</tr>
<tr>
<td>cfinput</td>
<td>Forms tags</td>
<td>Creates an input element (radio button, check box, text entry box); used in cfform</td>
</tr>
<tr>
<td>cfinsert</td>
<td>Database manipulation tags</td>
<td>Inserts records in a data source</td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cfinterface</td>
<td>Application framework tags,</td>
<td>Defines an interface that a ColdFusion component can implement</td>
</tr>
<tr>
<td></td>
<td>Extensibility tags</td>
<td></td>
</tr>
<tr>
<td>cfinvoke</td>
<td>Extensibility tags</td>
<td>Invokes component methods from a ColdFusion page or component</td>
</tr>
<tr>
<td>cfinvokeargument</td>
<td>Extensibility tags</td>
<td>Passes a parameter to a component method or a web service</td>
</tr>
<tr>
<td>cflayout</td>
<td>Display management tags</td>
<td>Creates a region of its container with a specific layout behavior</td>
</tr>
<tr>
<td>cflayoutarea</td>
<td>Display management tags</td>
<td>Defines a display region within a cflayout tag body</td>
</tr>
<tr>
<td>cfldap</td>
<td>Internet protocol tags</td>
<td>Provides access to LDAP directory servers</td>
</tr>
<tr>
<td>cflocation</td>
<td>Flow-control tags</td>
<td>Controls execution of a page</td>
</tr>
<tr>
<td>cflock</td>
<td>Application framework tags</td>
<td>Ensures data integrity and synchronizes execution of CFML code</td>
</tr>
<tr>
<td>cflog</td>
<td>Data output tags, Other tags</td>
<td>Writes a message to a log file</td>
</tr>
<tr>
<td>cflogin</td>
<td>Security tags</td>
<td>Defines a container for user login and authentication code</td>
</tr>
<tr>
<td>cfloginuser</td>
<td>Security tags</td>
<td>Identifies an authenticated user to ColdFusion</td>
</tr>
<tr>
<td>cflogout</td>
<td>Security tags</td>
<td>Logs the current user out</td>
</tr>
<tr>
<td>cfloop</td>
<td>Flow-control tags</td>
<td>Repeats a set of instructions based on conditions</td>
</tr>
<tr>
<td>cfmail</td>
<td>Communications tags, Internet</td>
<td>Assembles and posts an e-mail message</td>
</tr>
<tr>
<td></td>
<td>protocol tags</td>
<td></td>
</tr>
<tr>
<td>cfmailparam</td>
<td>Communications tags, Internet</td>
<td>Attaches a file or adds a header to an e-mail message</td>
</tr>
<tr>
<td></td>
<td>protocol tags</td>
<td></td>
</tr>
<tr>
<td>cfmailpart</td>
<td>Communications tags, Internet</td>
<td>Contains one part of a multipart mail message</td>
</tr>
<tr>
<td></td>
<td>protocol tags</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>cfmap</code></td>
<td>Other tags</td>
<td>Embeds a Google map within a ColdFusion web page</td>
</tr>
<tr>
<td><code>cfmapitem</code></td>
<td>Other tags</td>
<td>Creates markers on the map; a child tag of the <code>cfmap</code> tag</td>
</tr>
<tr>
<td><code>cfmediaplayer</code></td>
<td>Other tags</td>
<td>Creates an in-built media player that can play FLV files</td>
</tr>
<tr>
<td><code>cfmenu</code></td>
<td>Display management tags</td>
<td>Creates a top-level menu or a tool bar.</td>
</tr>
<tr>
<td><code>cfmenuItem</code></td>
<td>Display management tags</td>
<td>Defines an entry in a menu, including an item that is the head of a submenu.</td>
</tr>
<tr>
<td><code>cfmessagebox</code></td>
<td>Application framework tags</td>
<td>Defines a control for displaying pop-up messages</td>
</tr>
<tr>
<td><code>cfNTauthenticate</code></td>
<td>Security tags</td>
<td>Authenticates user information against an NT domain</td>
</tr>
<tr>
<td><code>cfoauth</code></td>
<td>Communications tags, Internet protocol tags</td>
<td>The <code>&lt;oauth&gt;</code> tag allows you to easily integrate third-party OAuth 2 authentication provider</td>
</tr>
<tr>
<td><code>cfobject</code></td>
<td>Extensibility tags</td>
<td>Creates COM, component, CORBA, Java, and web service objects</td>
</tr>
<tr>
<td><code>cfobjectcache</code></td>
<td>Database manipulation tags</td>
<td>Flushes the query cache</td>
</tr>
<tr>
<td><code>cfoutput</code></td>
<td>Data output tags</td>
<td>Displays the output of a database query or other operation</td>
</tr>
<tr>
<td><code>cfparam</code></td>
<td>Variable manipulation tags</td>
<td>Defines a parameter and its default value</td>
</tr>
<tr>
<td><code>cfpdf</code></td>
<td>Forms tags</td>
<td>Manipulates existing PDF documents.</td>
</tr>
<tr>
<td><code>cfpdfform</code></td>
<td>Forms tags</td>
<td>Creates and manipulates PDF forms.</td>
</tr>
<tr>
<td><code>cfpdfformparam</code></td>
<td>Forms tags</td>
<td>Creates interactive fields on a PDF form.</td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>cfpdfparam</code></td>
<td>Forms tags</td>
<td>Child tag of the <code>cfpdf</code> tag. Used only with the merge action to merge multiple pages or PDF documents into one file</td>
</tr>
<tr>
<td><code>cfpdfsubform</code></td>
<td>Forms tags</td>
<td>Creates subforms within a PDF form.</td>
</tr>
<tr>
<td><code>cfpod</code></td>
<td>Display management tags</td>
<td>Creates a an area of the browser or layout area with an optional title bar and a body</td>
</tr>
<tr>
<td><code>cfpop</code></td>
<td>Communications tags, Internet protocol tags</td>
<td>Gets and deletes messages from POP mail server</td>
</tr>
<tr>
<td><code>cfpresentation</code></td>
<td>Data output tags</td>
<td>Creates a presentation dynamically from an HTML page or SWF files</td>
</tr>
<tr>
<td><code>cfpresentationslide</code></td>
<td>Data output tags</td>
<td>Creates a slide dynamically from an HTML page or SWF source files (child tag of the <code>cfpresentation</code> tag)</td>
</tr>
<tr>
<td><code>cfpresenter</code></td>
<td>Data output tags</td>
<td>Describes a presenter in a slide presentation</td>
</tr>
<tr>
<td><code>cfprint</code></td>
<td>Data output tags</td>
<td>Prints PDF documents. Used for automated print jobs</td>
</tr>
<tr>
<td><code>cfprocessingdirective</code></td>
<td>Data output tags</td>
<td>Suppresses white space and other output</td>
</tr>
<tr>
<td><code>cfprocparam</code></td>
<td>Database manipulation tags</td>
<td>Holds parameter information for stored procedure</td>
</tr>
<tr>
<td><code>cfprocresult</code></td>
<td>Database manipulation tags</td>
<td>Result set name that ColdFusion tags use to access result set of a stored procedure</td>
</tr>
<tr>
<td><code>cfprogressbar</code></td>
<td>Other tags</td>
<td>Defines a progress bar to indicate the progress of an activity</td>
</tr>
<tr>
<td><code>cfproperty</code></td>
<td>Extensibility tags</td>
<td>Defines components</td>
</tr>
<tr>
<td><code>cfquery</code></td>
<td>Database manipulation tags</td>
<td>Passes SQL statements to a database</td>
</tr>
<tr>
<td><code>cfqueryparam</code></td>
<td>Database manipulation tags</td>
<td>Checks data type of a query parameter</td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>cfregistry</code></td>
<td>Other tags, Variable manipulation tags</td>
<td>Reads, writes, and deletes keys and values in a Windows system registry</td>
</tr>
<tr>
<td><code>cfreport</code></td>
<td>Exception handling tags</td>
<td>Embeds a ColdFusion Report Builder or Crystal Reports report</td>
</tr>
<tr>
<td><code>cfreportparam</code></td>
<td>Exception handling tags</td>
<td>Passes an input parameter to a ColdFusion Report Builder report</td>
</tr>
<tr>
<td><code>cfrethrow</code></td>
<td>Exception handling tags</td>
<td>Rethrows currently active exception</td>
</tr>
<tr>
<td><code>cfreturn</code></td>
<td>Extensibility tags</td>
<td>Returns results from a component method</td>
</tr>
<tr>
<td><code>cfsavecontent</code></td>
<td>Variable manipulation tags</td>
<td>Saves generated content inside tag body in a variable</td>
</tr>
<tr>
<td><code>cfschedule</code></td>
<td>Variable manipulation tags</td>
<td>Schedules page execution; optionally, produces static pages</td>
</tr>
<tr>
<td><code>cfscript</code></td>
<td>Application framework tags</td>
<td>Encloses a set of <code>cfscript</code> statements</td>
</tr>
<tr>
<td><code>cfsearch</code></td>
<td>Extensibility tags</td>
<td>Executes searches against data indexed in Solr collections, using <code>cfindex</code></td>
</tr>
<tr>
<td><code>cfselect</code></td>
<td>Forms tags</td>
<td>Creates a drop-down list box form element; used in <code>cfform</code> tag</td>
</tr>
<tr>
<td><code>cfset</code></td>
<td>Variable manipulation tags</td>
<td>Defines a variable</td>
</tr>
<tr>
<td><code>cfsetting</code></td>
<td>Other tags, Variable manipulation tags</td>
<td>Defines and controls ColdFusion settings</td>
</tr>
<tr>
<td><code>cfsharepoint</code></td>
<td>Extensibility tags</td>
<td>Invokes a SharePoint action from ColdFusion</td>
</tr>
<tr>
<td><code>cfsilent</code></td>
<td>Data output tags, Page processing tags</td>
<td>Suppresses CFML output within tag scope</td>
</tr>
<tr>
<td><code>cfslider</code></td>
<td>Forms tags</td>
<td>Creates slider control; used in <code>cfform</code></td>
</tr>
<tr>
<td><code>cfspreadsheet</code></td>
<td>Extensibility tags</td>
<td>Manages Excel spreadsheet files</td>
</tr>
<tr>
<td><code>cfsprydataset</code></td>
<td>Internet protocol tags</td>
<td>Creates a spry data set</td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cfstoredproc</td>
<td>Database manipulation tags</td>
<td>Holds database connection information; identifies a stored procedure to execute</td>
</tr>
<tr>
<td>cfswitch</td>
<td>Flow-control tags</td>
<td>Evaluates passed expression; passes control to matching cfcase tag</td>
</tr>
<tr>
<td>cftable</td>
<td>Data output tags</td>
<td>Builds a table in a ColdFusion page</td>
</tr>
<tr>
<td>cftextarea</td>
<td>Forms tags</td>
<td>Puts a multiline text box in a form</td>
</tr>
<tr>
<td>cfthread</td>
<td>Application framework tags</td>
<td>Creates and manages ColdFusion threads, independent streams of execution.</td>
</tr>
<tr>
<td>cfthrow</td>
<td>Exception handling tags, Flow-control tags</td>
<td>Throws a developer-specified exception</td>
</tr>
<tr>
<td>cftimer</td>
<td>Debugging tags</td>
<td>Displays execution time for a block of code</td>
</tr>
<tr>
<td>cftooltip</td>
<td>Display management tags</td>
<td>Specifies text to display when the mouse pointer hovers over the tag body elements</td>
</tr>
<tr>
<td>cftrace</td>
<td>Debugging tags</td>
<td>Displays and logs application debugging data</td>
</tr>
<tr>
<td>cftransaction</td>
<td>Database manipulation tags</td>
<td>Groups cfquery operations into one transaction; performs rollback processing</td>
</tr>
<tr>
<td>cftree</td>
<td>Forms tags</td>
<td>Creates tree control element; used in cfform</td>
</tr>
<tr>
<td>cftreeitem</td>
<td>Forms tags</td>
<td>Populates a tree control element in a form; used with cftree</td>
</tr>
<tr>
<td>cftry</td>
<td>Exception handling tags, Flow-control tags</td>
<td>Catches exceptions in ColdFusion pages</td>
</tr>
<tr>
<td>cfupdate</td>
<td>Database manipulation tags</td>
<td>Updates rows in a database data source</td>
</tr>
<tr>
<td>Tag</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cfwebsocket</td>
<td>Web Socket tags</td>
<td>Lets you create the WebSocket object in your CFM template. The tag creates a reference to the WebSocket JavaScript object at the client-side.</td>
</tr>
<tr>
<td>cfwddx</td>
<td>Extensibility tags</td>
<td>Serializes and deserializes CFML data structures to XML-based WDDX format</td>
</tr>
<tr>
<td>cfwindow</td>
<td>Display management tags</td>
<td>Creates a pop-up window in the browser</td>
</tr>
<tr>
<td>cfxml</td>
<td>Extensibility tags</td>
<td>Creates an XML document object</td>
</tr>
<tr>
<td>cfzip</td>
<td>File management tags</td>
<td>Manipulates ZIP and JAR files</td>
</tr>
<tr>
<td>cfzipparam</td>
<td>File management tags</td>
<td>Manipulates ZIP and JAR files</td>
</tr>
</tbody>
</table>

#back to top
Tags by function

This section lists tags by their function or purpose.
### Application framework tags

<table>
<thead>
<tr>
<th>cfapplication</th>
<th>cfimport</th>
<th>cfscript</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfassociate</td>
<td>cfinterface</td>
<td>cfthread</td>
</tr>
<tr>
<td>cferror</td>
<td>cflock</td>
<td></td>
</tr>
</tbody>
</table>
# Communications tags

<table>
<thead>
<tr>
<th>cfexchangecalendar</th>
<th>cfexchangefilter</th>
<th>cffeed</th>
<th>cfmailpart</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfexchangeconnectio n</td>
<td>cfexchangemail</td>
<td>cfmail</td>
<td>cfpop</td>
</tr>
<tr>
<td>cfexchangecontact</td>
<td>cfexchangetask</td>
<td>cfmailparam</td>
<td>cfimap</td>
</tr>
</tbody>
</table>
### Database manipulation tags

<table>
<thead>
<tr>
<th>cfdbinfo</th>
<th>cfprocparam</th>
<th>cfqueryparam</th>
<th>cfupdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfinsert</td>
<td>cfprocresult</td>
<td>cfquery</td>
<td></td>
</tr>
<tr>
<td>cfobjectcache</td>
<td>cfquery</td>
<td>cfstoredproc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cftransaction</td>
<td></td>
</tr>
</tbody>
</table>
## Data output tags

<table>
<thead>
<tr>
<th>cfchart</th>
<th>cfdocumentitem</th>
<th>cfpresentation</th>
<th>cfreportparam</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfchartdata</td>
<td>cfdocumentsection</td>
<td>cfpresentationslide</td>
<td>cfsilent</td>
</tr>
<tr>
<td>cfchartseries</td>
<td>cfheader</td>
<td>cfpresenter</td>
<td>cfetable</td>
</tr>
<tr>
<td>cfcol</td>
<td>cflog</td>
<td>cfprocessingdirective</td>
<td>cfreport</td>
</tr>
</tbody>
</table>
## Debugging tags

<table>
<thead>
<tr>
<th>cfdump</th>
<th>cftrace</th>
</tr>
</thead>
<tbody>
<tr>
<td>cftimer</td>
<td></td>
</tr>
</tbody>
</table>
## Display management tags

<table>
<thead>
<tr>
<th>cfdiv</th>
<th>cfmapitem</th>
<th>cfmenuitem</th>
<th>cfprogressbar</th>
</tr>
</thead>
<tbody>
<tr>
<td>cflayout</td>
<td>cfmediaplayer</td>
<td>cfmessagebox</td>
<td>cftooltip</td>
</tr>
<tr>
<td>cflayoutarea</td>
<td>cfmenu</td>
<td>cfpod</td>
<td>cfwindow</td>
</tr>
</tbody>
</table>
### Exception handling tags

<table>
<thead>
<tr>
<th>cfcatch</th>
<th>cffinally</th>
<th>cfthrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>cferror</td>
<td>cfrethrow</td>
<td>cftry</td>
</tr>
</tbody>
</table>
## Extensibility tags

<table>
<thead>
<tr>
<th>cfchart</th>
<th>cfchartdata</th>
<th>cfchartseries</th>
<th>cfcollection</th>
<th>cfcomponent</th>
<th>cfexecute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cfftp</td>
<td>cffunction</td>
<td>cfindex</td>
<td>cfinterface</td>
<td>cfinvoke</td>
</tr>
<tr>
<td></td>
<td>cfobject</td>
<td>cfproperty</td>
<td>cfinvokeargument</td>
<td>cfreport</td>
<td>cfreturn</td>
</tr>
<tr>
<td></td>
<td>cfsearch</td>
<td>cfsharepoint</td>
<td>cfwddx</td>
<td>cfwxml</td>
<td></td>
</tr>
</tbody>
</table>
File management tags

<table>
<thead>
<tr>
<th>cfdirectory</th>
<th>cfupload</th>
<th>cfzip</th>
</tr>
</thead>
<tbody>
<tr>
<td>cffile</td>
<td></td>
<td>cfzipparam</td>
</tr>
</tbody>
</table>
## Flow-control tags

<table>
<thead>
<tr>
<th>cfabort</th>
<th>cfelse</th>
<th>cfinclude</th>
<th>cfthrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfbreak</td>
<td>cfelseif</td>
<td>cflocation</td>
<td>cftry</td>
</tr>
<tr>
<td>cfcase</td>
<td>cfexecute</td>
<td>cfloop</td>
<td></td>
</tr>
<tr>
<td>cfcontinue</td>
<td>cfexit</td>
<td>cfrethrow</td>
<td></td>
</tr>
<tr>
<td>cfdefaultcase</td>
<td>cfif</td>
<td>cfswitch</td>
<td></td>
</tr>
</tbody>
</table>
## Forms tags

<table>
<thead>
<tr>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfapplet</td>
</tr>
<tr>
<td>cfcalendar</td>
</tr>
<tr>
<td>cffileupload</td>
</tr>
<tr>
<td>cfform</td>
</tr>
<tr>
<td>cfformgroup</td>
</tr>
<tr>
<td>cfformitem</td>
</tr>
<tr>
<td>cfgrid</td>
</tr>
<tr>
<td>cfgridcolumn</td>
</tr>
<tr>
<td>cfgridrow</td>
</tr>
<tr>
<td>cfgridupdate</td>
</tr>
<tr>
<td>cfinput</td>
</tr>
<tr>
<td>cfpdfform</td>
</tr>
<tr>
<td>cfpdfformparam</td>
</tr>
<tr>
<td>cfpdfparam</td>
</tr>
<tr>
<td>cfpdfsubform</td>
</tr>
<tr>
<td>cfselect</td>
</tr>
<tr>
<td>cfslider</td>
</tr>
<tr>
<td>cftextarea</td>
</tr>
<tr>
<td>cfTree</td>
</tr>
<tr>
<td>cfTreeitem</td>
</tr>
</tbody>
</table>
### Internet protocol tags

<table>
<thead>
<tr>
<th>cfajaximport</th>
<th>cfajaxproxy</th>
<th>cfhttp</th>
<th>cfhttpparam</th>
<th>cfldap</th>
<th>cfmail</th>
<th>cfmailparam</th>
<th>cfmailpart</th>
<th>cfpop</th>
<th>cfsprydataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>cffeed</td>
<td>cfimap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
# Page processing tags

<table>
<thead>
<tr>
<th>cfcache</th>
<th>cfheader</th>
<th>cfprocessingdirective</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfcontent</td>
<td>cfhtmlhead</td>
<td>cfsetting</td>
</tr>
<tr>
<td>cfflush</td>
<td>cfinclude</td>
<td>cfsilent</td>
</tr>
</tbody>
</table>
## Security tags

<table>
<thead>
<tr>
<th>cflogin</th>
<th>cfloginuser</th>
<th>cflogout</th>
<th>cfNTauthenticate</th>
</tr>
</thead>
</table>


## Variable manipulation tags

<table>
<thead>
<tr>
<th>cfcookie</th>
<th>cfparam</th>
<th>cfsavecontent</th>
<th>cfset</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfdump</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cfregistry</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cfschedule</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Web Socket tags

cfwebsocket
### Other tags

<table>
<thead>
<tr>
<th>cfimage</th>
<th>cflog</th>
<th>cfregistry</th>
</tr>
</thead>
</table>

[back to top]
Tag changes since ColdFusion 5

The following section lists tags, attributes, and values that have changed since ColdFusion 5, and indicate the specific release in which the change was made.
### New tags, attributes, and values

New and changed tags are also documented elsewhere, for the following releases:

- [New Tags in ColdFusion 9 (in the ColdFusion 9 version of this manual)](#)
- [Tags in ColdFusion 10](#)
- [New in ColdFusion 11](#)

This table lists tags, attributes, and attribute values that have been added since the ColdFusion MX release:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Attribute or value</th>
<th>Added in this ColdFusion release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple tags</td>
<td>attributeCollection</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfajaximport</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfajaxproxy</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfapplication</td>
<td>scriptProtect</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>loginStorage</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td>cfargument</td>
<td>component value of type attribute</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>xml value of type attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfcache</td>
<td>cachedirectory, timespan attributes</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfcalendar</td>
<td>onBlur and onFocus attributes</td>
<td>ColdFusion MX 7.01</td>
</tr>
<tr>
<td></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cfchart</td>
<td>style, title attributes</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>xAxisType, yAxisType attributes</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td></td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfchartdata</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfchartseries</td>
<td>datalabelstyle attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>horizontalbar value of type attribute</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td></td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>Tag</td>
<td>Attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>cfcollection</td>
<td>categories attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New values of the language attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>list and categoryList values of action attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>name attribute</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfcomponent</td>
<td>implements, serviceaddress attributes</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>component value of extends attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>style, namespace, serviceporttypename, wsdlfile, bindingname, and output attributes</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>Extended functionality for the hint and displayname attributes when publishing document-literal style web services</td>
<td></td>
</tr>
<tr>
<td></td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfcontent</td>
<td>variable attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cfcontinue</td>
<td>All</td>
<td>ColdFusion 9</td>
</tr>
<tr>
<td>cfdbinfo</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfdirectory</td>
<td>listinfo and type attributes</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>recurse attribute for list and delete actions</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cfdiv</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Tag</td>
<td>Attributes</td>
<td>ColdFusion Requires</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>cfdocument</td>
<td>bookmark, authPassword, authUser, localUrl, proxyHost, proxyPassword, proxyPort, proxyUser, saveAsName and userAgent attributes, totalsectionpagecount and currentsectionpagenumber scope variables</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>src, srcfile, and mimetype attributes</td>
<td>ColdFusion MX 7.01</td>
</tr>
<tr>
<td></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cfdocumentitem</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cfdocumentsection</td>
<td>name, authPassword, authUser, and userAgent attributes</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cfdump</td>
<td>show, format, hide, keys, meta info, output, and showUDFs attributes</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfexchangecalendar</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfexchangeconnection</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfexchangecontact</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfexchangefilter</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfexchangemail</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfexchangetask</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfexecute</td>
<td>variable attribute</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td>cffeed</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cffile</td>
<td>result attribute for action=&quot;upload&quot; action</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>fixnewline attribute for action= &quot;append&quot; and action= &quot;write&quot; actions</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>Tag Name</td>
<td>Version</td>
<td>Attributes and Features</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cffileupload</td>
<td>All</td>
<td>ColdFusion 9</td>
</tr>
<tr>
<td>cffinally</td>
<td>All</td>
<td>ColdFusion 9</td>
</tr>
<tr>
<td>cfform</td>
<td>onSuccess</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>onError</td>
<td></td>
</tr>
<tr>
<td></td>
<td>name, action</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>attributes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>accessible</td>
<td></td>
</tr>
<tr>
<td></td>
<td>format, height, width, method, onError, onReset, preloader, scriptsDir, skin, style, timeout, wMode</td>
<td>attributes</td>
</tr>
<tr>
<td>cfformgroup</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cfformitem</td>
<td>script, value of type</td>
<td>ColdFusion MX 7.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfftp</td>
<td>fingerprint, key, paraphrase, and secure attributes, quote, site, allo, and acct values to the action attribute</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>result</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cffunction</td>
<td>description</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>attribute; the XML value to the returntype attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfgrid</td>
<td>bind, bindOnLoad, pageSize, preservePageOnSort, stripeRows, stripeRowColor attributes, and HTML value of format attribute</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>onBlur, onFocus</td>
<td>ColdFusion MX 7.01</td>
</tr>
<tr>
<td></td>
<td>format attribute and support for Flash and XML output enabled, onchange, style, too ltip, visible attributes (Flash format only)</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>Tag</td>
<td>Attribute Description</td>
<td>Compatibility</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>cfgridcolumn</td>
<td>mask attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>currency value of type attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cfhttp</td>
<td>clientCert and clientCertPassword attributes</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>never value of GetAsBinary attribute</td>
<td>ColdFusion MX 7.01</td>
</tr>
<tr>
<td></td>
<td>result attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>HEAD, PUT, DELETE, OPTIONS, and TRACE values of method attribute</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td></td>
<td>multipart, getasbinary, proxyUser, proxyPassword attributes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>charset, firstrowasheaders attributes</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfhtpparam</td>
<td>header and body values of type attribute</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td></td>
<td>encoded, mimeType attributes</td>
<td></td>
</tr>
<tr>
<td>cfimage</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfimport</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfimap</td>
<td>All</td>
<td>ColdFusion 9</td>
</tr>
<tr>
<td>cfindex</td>
<td>prefix attribute</td>
<td>ColdFusion MX 7.01</td>
</tr>
<tr>
<td></td>
<td>custom3, custom4, category, and categorytree attributes for update and refresh actions</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>status attribute for update, refresh, delete, and purge actions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New values of the language attribute</td>
<td></td>
</tr>
<tr>
<td>cfinput</td>
<td>autosuggest, autosuggestBin, dDelay, autosuggestMinLength, delimiter, maxResultsDisplayed, showAutosuggestLoadingIcon, sourceForTooltip, and typeahead attributes.</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td>support for the bind attribute in HTML forms and the bindAttribute, bindOnload, and onBindErr or attributes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>datefield value of the type attribute in HTML forms</td>
<td></td>
</tr>
<tr>
<td></td>
<td>height and width attributes (all except checkbox and radiobutton)</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>bind attribute (text and password)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>label attribute (all but button, image, reset, and submit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mask attribute (text only)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>validateAt attribute (all but button, image, reset, and submit)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>datefield, button, file, hidden, image, reset, and submit values of type attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>daynames and monthnames attributes (type=&quot;datefield&quot; only)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>boolean, email, guid, maxLength, noblanks, range, submitOnce, URL, USdate, and uuid values of the validate attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tooltip, visible, and enabled attributes (Flash forms only)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>src attribute (image only)</td>
<td></td>
</tr>
<tr>
<td>cfinterface</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Tag</td>
<td>Description</td>
<td>Compatibility</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>cfinvoke</td>
<td>refreshWSDL, wsd2java arguments</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>servicePort</td>
<td>attribute for web services</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>All</td>
<td></td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfinvokeargument</td>
<td>omitt attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>All</td>
<td></td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cflayout</td>
<td></td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cflayoutarea</td>
<td></td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfldap</td>
<td>returnAsBinary attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cflocation</td>
<td>statusCode attribute</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cflock</td>
<td>Request value of scope attribute</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cflogin</td>
<td></td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfloginuser</td>
<td></td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cflogout</td>
<td></td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfloop</td>
<td>characters, file, and array attributes</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfmail</td>
<td>priority, useSSL, and useTLS</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfmailparam</td>
<td>contentID, {{disposition}} attributes</td>
<td>ColdFusion MX 7</td>
</tr>
</tbody>
</table>

The `cfmail` tag no longer lets you send multipart mail by embedding the entire MIME-encoded message in the tag body. Use the `cfmailpart` tag, instead.

charset, failto, replyTo, userName, password, wrapText attributes

spoolEnable attribute

ColdFusion MX 6.1
<table>
<thead>
<tr>
<th>type</th>
<th>attribute</th>
<th>ColdFusion MX 6.1</th>
<th>ColdFusion MX 6.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfmailpart</td>
<td>All</td>
<td>ColdFusion MX 6.1</td>
<td></td>
</tr>
<tr>
<td>cfmap</td>
<td>All</td>
<td>ColdFusion 9</td>
<td></td>
</tr>
<tr>
<td>cfmapitem</td>
<td>All</td>
<td>ColdFusion 9</td>
<td></td>
</tr>
<tr>
<td>cfmediaplayer</td>
<td>All</td>
<td>ColdFusion 9</td>
<td></td>
</tr>
<tr>
<td>cfmenu</td>
<td>All</td>
<td>ColdFusion 8</td>
<td></td>
</tr>
<tr>
<td>cfmenuitem</td>
<td>All</td>
<td>ColdFusion 8</td>
<td></td>
</tr>
<tr>
<td>cfmessagebox</td>
<td>All</td>
<td>ColdFusion 9</td>
<td></td>
</tr>
<tr>
<td>cfNTauthenticate</td>
<td>All</td>
<td>ColdFusion MX 7</td>
<td></td>
</tr>
<tr>
<td>cfobject</td>
<td>.net value of type attribute and related assembly, port, protocol, and secure attributes</td>
<td>ColdFusion 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>password, proxyPassword, proxyPort, proxyServer, proxyUser, refreshWSDL, userName, wsdl2JavaArgs, and wsportname attributes for web services</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>component and webservice attributes</td>
<td>ColdFusion MX</td>
<td></td>
</tr>
<tr>
<td>cfoobjectcache</td>
<td>All</td>
<td>ColdFusion MX</td>
<td></td>
</tr>
<tr>
<td>cfpparam</td>
<td>min, max, pattern attributes</td>
<td>ColdFusion MX 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>creditcard, email, eurodate, float, integer, range, regex, regular_expression, ssn, social_security_number, time, URL, USdate, XML, zipcode values of the type attribute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>Description</td>
<td>Compatibility</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
<td>---------------</td>
<td></td>
</tr>
</tbody>
</table>
| cfpdf | action = "thumbnail" has the following new attributes:  
  - hires  
  - compressiffs  
  - maxlength  
  - maxbreadth  
  - maxscale  
  - action = "optimize" (new action with all new attributes)  
  - action = "addheader" (new action)  
  - action = "addfooter" (new action)  
  - action = "removeheaderfooter" (new action)  
  - action = "extracttext"  
  - action = "extractimage"  
  - action = "write" package = "true"  
  - action = "merge"  
  - encodeall="true"  
  - action = "write" name=#PDF variable#  
  - action = "transform" (new action) | ColdFusion 9 |
<p>| cfpdfform | All | ColdFusion 8 |
| cfpdfformparam | All | ColdFusion 8 |
| cfpdfparam | All | ColdFusion 8 |
| cfpdfsubform | All | ColdFusion 8 |
| cfpod | All | ColdFusion 8 |
| cfpop | cids query variable | ColdFusion MX 7.01 |
| cfpresentation | All | ColdFusion 8 |
| cfpresentationslide | All | ColdFusion 8 |
| cfpresenter | All | ColdFusion 8 |
| cfprint | All | ColdFusion 8 |
| cfprocessingdirective | pageEncoding attribute | ColdFusion MX |</p>
<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfprocparam</td>
<td></td>
<td>Cold Fusion 9</td>
</tr>
<tr>
<td>cfprogressbar</td>
<td></td>
<td>Cold Fusion 9</td>
</tr>
<tr>
<td>cfproperty</td>
<td></td>
<td>Cold Fusion MX</td>
</tr>
<tr>
<td>cfquery</td>
<td>result attribute</td>
<td>Cold Fusion MX 7</td>
</tr>
<tr>
<td>cfreturn</td>
<td></td>
<td>Cold Fusion MX</td>
</tr>
<tr>
<td>cfreport</td>
<td>HTML, XML values of format attribute, resourceTimespan, style attributes</td>
<td>Cold Fusion 8</td>
</tr>
<tr>
<td></td>
<td>RTF value of format attribute</td>
<td>Cold Fusion MX 7.01</td>
</tr>
<tr>
<td></td>
<td>template, format, name, filename, query, overwrite attribute</td>
<td>Cold Fusion MX 7</td>
</tr>
<tr>
<td>cfreportparam</td>
<td>chart, query, series, style, subreport attributes</td>
<td>Cold Fusion 8</td>
</tr>
<tr>
<td>name, value attributes</td>
<td></td>
<td>Cold Fusion MX 7</td>
</tr>
<tr>
<td>cfsearch</td>
<td>category, categoryTree, status, suggestions, contextPassages, contextBytes, contextHighlightBegin, contextHighlightEnd, previousCriteria attributes</td>
<td>Cold Fusion MX 7</td>
</tr>
<tr>
<td></td>
<td>natural, internet, and internet_basic values of type attribute</td>
<td></td>
</tr>
<tr>
<td>cfselect</td>
<td>support for the bind attribute in HTML forms and the bindAttribute, bindOnload, and onBindError attributes.</td>
<td>Cold Fusion 8</td>
</tr>
<tr>
<td></td>
<td>Support for tooltips in HTML forms including the sourceForTooltip attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>selected attribute can take a list</td>
<td>Cold Fusion MX 7</td>
</tr>
</tbody>
</table>
enabled, group, height, label, onKeyUp, onKeyDown, onMouseUp, onMouseDown, onChange, onClick, queryPosition, tooltip, visible, and width attributes

<table>
<thead>
<tr>
<th>cfsetting</th>
<th>requestTimeOut attribute</th>
<th>ColdFusion MX</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfsharepoint</td>
<td>All</td>
<td>ColdFusion 9</td>
</tr>
<tr>
<td>cfspreadsheet</td>
<td>All</td>
<td>ColdFusion 9</td>
</tr>
<tr>
<td>cfsprydataset</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfstoredproc</td>
<td>result attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cftextarea</td>
<td>Rich text editor support including the following attributes (HTML format only): richtext, basepath, fontFormats, fontNames, fontSizes, skin, stylesXML, templatesXML, toolbar, toolbarOnFocus, and support for the height and width attributes in HTML format</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td></td>
<td>support for the bind attribute and bindAttribute, bindOnLoad, and onBindError attributes in HTML format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>support for tooltips in HTML format including tooltip and sourceForToolTip attribute</td>
<td></td>
</tr>
<tr>
<td>html attribute</td>
<td>ColdFusion MX 7.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cfthread</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfthrow</td>
<td>object attribute</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cftimer</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cftooltip</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cftree</td>
<td>onBlur and onFocus attributes</td>
<td>ColdFusion MX 7.01</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td></td>
<td>format, onChange, style attributes</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cftrace</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfwindow</td>
<td>ALL</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfxml</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfzip</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>cfzipparam</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
</tbody>
</table>
### Deprecated tags, attributes, and values

The following tags, attributes, and attribute values are deprecated. Do not use them in ColdFusion applications. They might not work, and might cause an error, in releases later than ColdFusion MX.

#### Updated information for ColdFusion 11

Please note that there is a new document for ColdFusion 11, located outside of this manual, with additional and more details information about deprecated and removed/obsoleted features. See [Deprecated Features](#).

<table>
<thead>
<tr>
<th>Tag</th>
<th>Attribute or value</th>
<th>Deprecated as of this ColdFusion release</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cfcache</code></td>
<td>cachedirectory, timeout attributes</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>cfcollection</code></td>
<td>map and repair options of the action attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><code>cferror</code></td>
<td>monitor option of the exception attribute</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>cffile</code></td>
<td>system value for attributes attribute</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td></td>
<td>temporary value for attributes attribute</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>cfform</code></td>
<td>passthrough attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td></td>
<td>enableCAB attribute</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>cfftp</code></td>
<td>agentname attribute</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>cfgraph</code></td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>cfgraphdata</code></td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>cffgridupdate</code></td>
<td>connectString, dbName, dbServer, dbType, provider, providerDSN attributes</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>cfinput</code></td>
<td>passthrough attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><code>cfinsert</code></td>
<td>connectString, dbName, dbServer, dbType, provider, providerDSN attributes</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>Tag</td>
<td>Description</td>
<td>Version</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>cfldap</td>
<td>The <code>filterFile</code> attribute is supported.</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cflog</td>
<td>The <code>date</code>, <code>thread</code>, <code>time</code> attributes are supported.</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfquery</td>
<td>The <code>connectString</code>, <code>dbName</code>, <code>dbServer</code>, <code>provider</code>, <code>providerDSN</code>, <code>sql</code> attributes are supported.</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td></td>
<td><strong>The following <code>dbType</code> attribute values:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>dynamic</code>, <code>ODBC</code>, <code>Oracle73</code>, <code>Oracle80</code>, <code>Sybase11</code>, <code>OLEDB</code>, <code>DB2</code></td>
<td></td>
</tr>
<tr>
<td>cfregistry</td>
<td>All, on UNIX only</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfsearch</td>
<td>The <code>external</code>, <code>language</code> attributes are supported.</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfselect</td>
<td>The <code>passthrough</code> attribute is supported.</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cf servlet</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cf servlet param</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cf slider</td>
<td><code>img</code>, <code>imgStyle</code>, <code>grooveColor</code>, <code>refreshLabel</code>, <code>tickmarkimages</code>, <code>tickmarklabels</code>, <code>tickmarkmajor</code>, <code>tickmarkminor</code> attributes are supported.</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cf stored proc</td>
<td>The <code>connectString</code>, <code>dbName</code>, <code>dbServer</code>, <code>dbtype</code>, <code>provider</code>, <code>providerDSN</code> attributes are supported.</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cf text input</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cf update</td>
<td>The <code>connectString</code>, <code>dbName</code>, <code>dbServer</code>, <code>dbtype</code>, <code>provider</code>, <code>providerDSN</code> attributes are supported.</td>
<td>ColdFusion MX</td>
</tr>
</tbody>
</table>
# Obsolete tags, attributes, and values

The following tags, attributes, and attribute values are obsolete. Do not use them in ColdFusion applications. They do not work, and might cause an error, in releases later than ColdFusion 5.

## Updated information for ColdFusion 11

Please note that there is a new document for ColdFusion 11, located outside of this manual, with additional and more details about deprecated and removed/obsoleted features. See [Deprecated Features](#).

<table>
<thead>
<tr>
<th>Tag</th>
<th>Attribute or value</th>
<th>Obsolete as of this ColdFusion release</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfauthenticate</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfchart</td>
<td>rotated attribute</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>cffile</td>
<td>attributes attribute value archive</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfimpersonate</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfindex</td>
<td>action attribute value optimize</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td></td>
<td>external attribute</td>
<td></td>
</tr>
<tr>
<td>cfinternaladminsecurity</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This tag did not appear in CFML Reference.</td>
</tr>
<tr>
<td>cfldap</td>
<td>filterConfig and filterFile attributes</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>cfnewinternaladminsecurity</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This tag did not appear in CFML Reference.</td>
</tr>
<tr>
<td>cfsetting</td>
<td>catchExceptionsByPattern attribute</td>
<td>ColdFusion MX</td>
</tr>
</tbody>
</table>
This section lists tags starting with alphabets A-B with their description.

- cfabort
- cfajaximport
- cfajaxproxy
- cfapplet
- cfapplication
- cfargument
- cfassociate
- cfauthenticate
- cfbreak
**cfabort**

**Description**

Stops the processing of a ColdFusion page at the tag location. ColdFusion returns everything that was processed before the tag. The tag is often used with conditional logic to stop processing a page when a condition occurs.

**Category**

*Flow-control tags*

**Syntax**

```plaintext
<cfabort
    showError = "error message">
```

**Note**

You can specify this tag's attributes in an attributeCollection whose value is a structure. Specify the structure name in the attributeCollection and use the tag's attribute names as structure keys.

**See also**

cfbreak, cfexecute, cfexit, cfif, cflocation, cfloop, cfswitch, cfthrow, cftry; cfabort and cfexit in the Developing ColdFusion Applications

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showError</td>
<td>Optional</td>
<td></td>
<td>Error to display, in a standard ColdFusion error page, when tag executes.</td>
</tr>
</tbody>
</table>

**Usage**

When you use the `cfabort` and `cferror` tags together, the `cfabort` tag halts processing immediately; the `cferror` tag redirects output to a specified page. If this tag does not contain a `showError` attribute value, processing stops when the tag is reached and ColdFusion returns the page contents up to the line that contains the `cfabort` tag.

When you use this tag with the `showError` attribute, but do not define an error page using `cferror`, page processing stops when the `cfabort` tag is reached. The message in `showError` displays to the client. When you use this tag with the `showError` attribute and an error page using `cferror`, ColdFusion redirects output to the error page specified in the `cferror` tag.

**Note**

When using `cfabort`, `cflocation`, or `cfcontent` tags, the `OnAbort` method is invoked instead on `OnRequestEnd`. 
Example

This example shows the use of `cfabort` to stop processing. In the second example, where `cfabort` is used, the result never displays.

```xml
<h3>Example A: Let the instruction complete itself</h3>
<!---- first, set a variable ---->
<cfset myVariable = 3>
<!---- now, perform a loop that increments this value ---->
<cfloop from = "1" to = "4" index = "Counter">
<cfset myVariable = myVariable + 1>
</cfloop>

<cfoutput>
<p>The value of myVariable after incrementing through the loop #Counter# times is: #myVariable#</p>
</cfoutput>

<h3>Example B: Use `cfabort` to halt the instructions with showmessage attribute and `cferror`</h3>
<!---- Reset the variable and show the use of `cfabort`. ---->
<cfset myVariable = 3>
<!---- Now, perform a loop that increments this value. ---->
<cfloop from = "1" to = "4" index = "Counter">
<!---- On the second time through the loop, `cfabort`. ---->
<cfif Counter is 2>
<!---- Take out the `cferror` line to see `cfabort` error processed by CF error page. ---->
<cferror type="request" template="request_err.cfm">
<cfabort showerror="CFABORT has been called for no good reason">
<!---- Processing is stopped, ---->
<!---- and subsequent operations are not carried out.---->
<cfelse>
<cfset myVariable = myVariable + 1>
</cfif>
</cfloop>

<cfoutput>
<p>The value of myVariable after incrementing through the loop#counter# times is: #myVariable#</p>
</cfoutput>
```
**cfajaximport**

**Description**

Controls the JavaScript files that are imported for use on pages that use ColdFusion AJAX tags and features.

**Category**

Internet protocol tags

**Syntax**

```xml
<cfajaximport
   cssSrc = "local URL path"
   params = "parameters"
   scriptSrc = "local URL path"
   tags = "comma-delimited list">
```

**Note**

You can specify this tag’s attributes in an `attributeCollection` whose value is a structure. Specify the structure name in the `attributeCollection` and use the tag’s attribute names as structure keys.

**See also**

cfform, cfgrid, cfinput, cflayout, cfmenu, cfpod, cfsprydataset, cftextarea, cftooltip, cftree, cfwindow. Specifying client-side support files in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>cssSrc</th>
<th>Optional</th>
<th>scriptsrc/ajax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the URL, relative to the web root, of the directory that contains the CSS files used by ColdFusion AJAX features, with the exception of the rich text editor. This directory must have the same directory structure, and contain the same CSS files, and image files required by the CSS files, as the <code>web_root/CFIDE/scripts/ajax/resources</code> directory. This attribute lets you create different custom styles for ColdFusion AJAX controls in different applications.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>params</th>
<th>Optional</th>
<th>This attribute allows you to specify parameters for the CFM page. Currently, the only parameter that you can specify is <code>googlemapkey</code>. You can specify the <code>googlemapkey</code> as follows: <code>&lt;cfajaximport params=#{googlemapkey=&quot;Map API Key&quot;}&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>scriptSrc</strong></td>
<td>Optional</td>
<td><strong>scriptSrc</strong> setting in the Administrator: default path is /CFIDE/scripts/</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| **tags** | Optional | A comma-delimited list of tags or tag-attribute combinations for which to import the supporting JavaScript files on this page. If you use this attribute, it must specify all ColdFusion AJAX tags that you use on the page and on any pages specified in `tagsource` attributes. For a list of valid attribute values and their purposes, see Usage. | Usage

**Using the `scriptSrc` and `cssSrc` attributes**

The `scriptSrc` attribute is useful if the JavaScript files are not in the default location. This attribute is required in some hosting environments and configurations that block access to the /CFIDE directory. The default `scriptSrc` value is determined by the Default CFFORM ScriptSrc Directory setting on the Server Settings > Settings page of the ColdFusion Administrator. For `cfform` tags, the tag's `scriptSrc` attribute takes precedence over this attribute.
You can use this attribute only if the `cfajaximport` tag is on a top-level page; that is, a page that the client directly requests. You cannot use it, for example, on a page that is specified in a `cfwindow` tag `source` attribute.

When you use the `cfajaximport` tag with a `scripts` attribute, the specified directory must have the same structure as the `/CFIDE/scripts` directory. For example, if you specify `scripts` as `"/resources/myScripts"`, the JavaScript files used by AJAX must be in the `/resources/myScripts/ajax` directory. This attribute specifies the folder that contains the ColdFusion client-side files for all subsequent tags on the current page, not just for AJAX-based tags. Therefore, the directory tree must include all ColdFusion client-side files used by those tags. For example, if a `cform` tag on the page is in Flash or applet format, include the CF_RunActiveContent.js file in the directory specified by the `scripts` attribute.

You use the `cssSrc` attribute to specify the location of the CSS files required by ColdFusion AJAX features. This attribute overrides the `/ajax/resources` directory for the current page. Therefore, if all pages that use a custom `scripts` directory also use a custom `cssSrc` directory, you do not have to include the ColdFusion AJAX CSS files in the `scripts` directory tree.

**Using the `tags` attribute or no attribute**

If you do not use the `cfajaximport` tag on a page that contains ColdFusion tags with AJAX UI features, ColdFusion correctly imports the required JavaScript files in most cases. Use this tag to explicitly import JavaScript files in these cases:

- If you use a ColdFusion AJAX JavaScript function, such as `ColdFusion.navigate`, `ColdFusion.Ajax.submitForm`, or `ColdFusion.Log.info` on a page that does not otherwise import any AJAX JavaScript functions, use the `cfajaximport` tag with no attribute to import the base JavaScript functions only. For example, use this tag on a page that does not include any ColdFusion AJAX-based tags,
  - If the following conditions are true:
    - You use any `source` attributes in `cflayoutarea`, `cfpod`, or `cfwindow` tags, or `bind` attribute in `cfdiv` tag.
    - The file that the `source` or `bind` attribute specifies has any of the tags listed in the following table.
    - You do not use each of the listed tags on the top-level page.

If these conditions are true, the top-level page must use the `cfajaximport` tag with a `tags` attribute that specifies the tags that only the other pages use. Otherwise, ColdFusion cannot identify that it will be using the tags and does not import the necessary JavaScript files.

You can specify any or all the following `tags` attribute values:

<table>
<thead>
<tr>
<th>Attribute value</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cfdiv</code></td>
<td><code>cfdiv</code> tags</td>
</tr>
<tr>
<td><code>cform</code></td>
<td>Forms that are in <code>cfpod</code>, <code>cfwindow</code>, or <code>cflayoutarea</code> tag bodies</td>
</tr>
<tr>
<td><code>cfgrid</code></td>
<td>AJAX format <code>cfgrid</code> tags</td>
</tr>
<tr>
<td><code>cfinput-autosuggest</code></td>
<td><code>cfinput</code> tags that use the <code>autosuggest</code> attribute</td>
</tr>
<tr>
<td><code>cfinput-datefield</code></td>
<td>HTML format <code>cfinput</code> tags that use the <code>datefield</code> attribute</td>
</tr>
<tr>
<td><code>cflayout-border</code></td>
<td><code>cflayout</code> tags with a <code>type</code> attribute value of <code>border</code></td>
</tr>
<tr>
<td>Tag</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>cflayout-tab</td>
<td>cflayout tags with a type attribute value of tab</td>
</tr>
<tr>
<td>cfmenu</td>
<td>cfmenu tags</td>
</tr>
<tr>
<td>cfpod</td>
<td>cfpod tags</td>
</tr>
<tr>
<td>cfsprydataset-JSON</td>
<td>cfspystdataset tags that generate Spry JSON data sets</td>
</tr>
<tr>
<td>cfsprydataset-XML</td>
<td>cfspystdataset tags that generate Spry XML data sets</td>
</tr>
<tr>
<td>cftextarea</td>
<td>HTML format cftextarea tags</td>
</tr>
<tr>
<td>cftooltip</td>
<td>cftooltip tags</td>
</tr>
<tr>
<td>cftree</td>
<td>HTML format cftree tags</td>
</tr>
<tr>
<td>cfwindow</td>
<td>cfwindow tags</td>
</tr>
</tbody>
</table>

**Example**

The following `cfajaximport` tag example specifies separate custom locations for the scripts used for AJAX features and for the AJAX CSS files. It also imports all JavaScript files used for `cftree`, and `cftooltip`.

```xml
<cfajaximport cssSrc="/collegeApp/application/cssFiles"
scriptsrc="/collegeApp/ajaxScripts"
tags="cftooltip, cfwindow">
```
**cfajaxproxy**

**Description**

Creates a JavaScript proxy for a ColdFusion component, for use in an AJAX client. Alternatively, creates a proxy for a single CFC method, JavaScript function, or URL that is bound to one or more control attribute values.

**Category**

Internet protocol tags

**Syntax**

```cfc
<cfajaxproxy
  cfc = "CFC name"
  jscclassname = "JavaScript proxy class name">

OR

<cfajaxproxy
  bind = "bind expression"
  onError = "JavaScript function name"
  onSuccess = "JavaScript function name">
```

**Note**

You can specify this tag’s attributes in an attributeCollection whose value is a structure. Specify the structure name in the attributeCollection and use the tag’s attribute names as structure keys.

**See also**

DeserializeJSON, IsJSON, SerializeJSON, Using Ajax Data and Development Features in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added this tag

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>bind</th>
<th>A bind or cfc attribute is required</th>
<th>A bind expression that specifies a CFC method, JavaScript function, or URL to call. For detailed information about specifying bind expressions, see <a href="#">Binding data to form fields</a> in the Developing ColdFusion Applications. You cannot use this attribute with the <code>cfc</code> attribute.</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfc</td>
<td>A bind or cfc attribute is required</td>
<td>The CFC for which to create a proxy. Specify a dot-delimited path to the CFC. The path can be an absolute filepath, or relative to location of the CFML page. For example, if the myCFC CFC is in the cfcs subdirectory of the ColdFusion page, specify cfcs.myCFC. If the CFC extends another CFC, the extended CFC methods are also available to the JavaScript proxy. On UNIX-based systems, the tag searches first for a file whose name or path corresponds to the specified name or path, but is in all lowercase. If it does not find it, ColdFusion then searches for a filename or path that corresponds to the attribute value exactly, with identical character casing. This attribute cannot be used with the <code>bind</code> attribute.</td>
</tr>
<tr>
<td>jsclassname</td>
<td>Optional</td>
<td>The value of the cfc attribute</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>onError</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>onSuccess</td>
<td>Optional</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

Make sure that the Enable HTTP Status Codes option on the Server Settings > Settings page of the ColdFusion Administrator is selected. Otherwise, the proxy cannot determine if an error occurs in invoking the CFC function and cannot call the error handler. A cfajaxproxy tag with a bind attribute does not refresh any control that is not specified in the bind expression.

If you specify a URL in the bind attribute, the HTTP response must consist of a single JSON representation of an object, array, or simple value, corresponding to the onSuccess argument.

**Creating a CFC proxy**

The cfajaxproxy tag with a cfc attribute generates a JavaScript proxy that represents a CFC on the web client. The tag and the proxy it generates have the following characteristics:

- The proxy provides one function that corresponds to each CFC remote function. Calling these functions in your client-side JavaScript code remotely calls the CFC functions on the server.
- The proxy provides several functions that you call to configure the interaction between the client and the server. These functions set the HTTP method and synchronization mode of the XMLHttpRequest call that the proxy uses to interact with the server. The functions also can specify a JavaScript callback handler and an
error handler for asynchronous calls.

- Because JavaScript is case sensitive, ensure that you match the case of the keys in any ColdFusion structure or scope that you send to the client. By default, ColdFusion sets variable names and structure element names to all-uppercase. (You can create structure element names with lowercase characters by specifying the names in associative array notation, for example, myStruct"myElement"="value".) The keys for the two arrays in the JSON object that the ColdFusion serializeJSON tag generates to represent a query are COLUMNS and DATA, for example, not columns and data.

For detailed information on using AJAX CFC proxies, see Using ColdFusion Ajax CFC proxies in Using Ajax Data and Development Features in the Developing ColdFusion Applications.

- **Note**

  The proxy passes a _CF_NODEBUG Boolean argument to called CFC functions. ColdFusion checks this value, and when it is true, does not append to the response any debugging information that it normally would send. This behavior ensures that the JSON responses to AJAX requests do not include any non-JSON (i.e., debug information) text.

CFC proxy utility functions

When you use the cfc option, the JavaScript proxy object provides the following functions that you can use to control interaction with the server:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setAsyncMode()</td>
<td>Sets the call mode to asynchronous. The calling thread (the Java thread of the client system that is processing the page) is not blocked when you make a call to a proxy function, so page processing can continue while waiting for a response from the server. The proxy invokes the function specified by the setCallbackHandler function with the response from the server. If an error occurs, the proxy invokes the error handler specified by the setErrorHandler function.</td>
</tr>
<tr>
<td>setCallbackHandler(function)</td>
<td>Specifies the callback handler for an asynchronous call. The function parameter is the JavaScript function to invoke as an argument. The callback function must take one parameter, the return value from the CFC that the proxy has deserialized from JSON to a JavaScript representation. This method automatically sets the call mode to asynchronous.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| `setErrorHandler(function)` | Sets the error handler that the proxy invokes if there is an error in an asynchronous call. The `function` parameter is the JavaScript function to invoke. The error handler function must take two parameters:  
- An HTTP error code  
- A status message  
  This method automatically sets the call mode to asynchronous. |
| `setForm(ID)` | Adds names and values of the fields in the form specified by the `ID` attribute to the arguments passed by a proxy function that is called immediately after this function.  
For more information, see Submitting data to a CFC in the Developing ColdFusion Applications. |
| `setHTTPMethod("method")` | Sets the HTTP method to use for the call. The `method` parameter is a case-insensitive string, and must have one of the following values:  
- GET (the default method)  
- POST |
| `setQueryFormat(format)` | Specifies the JSON format in which to return ColdFusion query data. The parameter must have one of the following values:  
- `row`: (Default) Sends the data as a JSON object with two entries: the column names and an array of row arrays.  
- `column`: Sends the data as a JSON object that represents WDDX query format. This object has three entries: the number of rows, an array with the column names, and an object where the keys are the column names and the values are arrays containing the column data.  
For more information on query formats, see [SerializeJSON](http://example.com). |
setReturnFormat(format)

Specifies the format in which the CFC function returns the result. ColdFusion automatically converts the function return value into the specified format before returning it to the client. The parameter must have one of the following values:

- `json` (the default format if you don’t use this function)
- `plain`
- `wddx`

If you specify `plain`, and set the “content-type” header on the response from the server to `text/xml`, the proxy returns an XML object to the caller or callback function. If the content type is not set to `text/xml`, the return value from the server is returned as-is. This function is useful if you return XML or a plain string to the browser.

setSyncMode()

Sets the call mode to synchronous (the default synchronization mode). The calling thread remains blocked until the call returns. If an error occurs, the proxy throws an exception. In synchronous mode, the methods in the CFC proxy return the CFC method results directly to the caller.

Example

The following example uses a remote CFC method to populate a drop-down list of employees. When you select a name from the list, it uses a call to the CFC method to get information about the employee, and displays the results. The application page has the following lines:

```html
<!---- The cfajaxproxy tag creates a client-side proxy for the emp CFC.
View the generated page source to see the resulting JavaScript.
The emp CFC is in the components subdirectory of the directory that contains this page. --->
<cfajaxproxy cfc="components.emp" jsclassname="emp">

<html>
<head>
<script type="text/javascript">

// Function to find the index in an array of the first entry
// with a specific value.
// It is used to get the index of a column in the column list.
Array.prototype.findIndex = function(value) {
  for (var i=0; i < this.length; i++) {
    if (this[i] == value) {
      return i;
    }
  }
}

// Use an asynchronous call to get the employees for the
```
// drop-down employee list from the ColdFusion server.
var getEmployees = function(){
    // create an instance of the proxy.
    var e = new emp();
    // Setting a callback handler for the proxy automatically makes
    // the proxy's calls asynchronous.
    e.setCallbackHandler(populateEmployees);
    e.setErrorHandler(myErrorHandler);
    // The proxy getEmployees function represents the CFC
    // getEmployees function.
    e.getEmployees();
}

// Callback function to handle the results returned by the
// getEmployees function and populate the drop-down list.
var populateEmployees = function(res)
{
    with(document.simpleAJAX){
        var option = new Option();
        option.text='Select Employee';
        option.value='0';
        employee.options[0] = option;
        for(i=0;i<res.DATA.length;i++){
            var option = new Option();
            option.text=res.DATA[i][res.COLUMNS.findIdx('FIRSTNAME')]
            + ' ' + res.DATA[i][res.COLUMNS.findIdx('LASTNAME')];
            option.value=res.DATA[i][res.COLUMNS.findIdx('EMP_ID')];
            employee.options[i+1] = option;
        }
    }
}

// Use an asynchronous call to get the employee details.
// The function is called when the user selects an employee.
var getEmployeeDetails = function(id){
    var e = new emp();
    e.setCallbackHandler(populateEmployeeDetails);
    e.setErrorHandler(myErrorHandler);
    // This time, pass the employee name to the getEmployees CFC
    // function.
    e.getEmployees(id);
}

// Callback function to display the results of the getEmployeeDetails
// function.
var populateEmployeeDetails = function(employee)
{
    var eId = employee.DATA[0][0];
    var efname = employee.DATA[0][1];
    var elname = employee.DATA[0][2];
    var eemail = employee.DATA[0][3];
    var ephone = employee.DATA[0][4];
    var edepartment = employee.DATA[0][5];

    with(document.simpleAJAX){
        empData.innerHTML =
        '<span style="width:100px">Employee Id:</span>'
        + '<span style="color:green">Employee Id:</span>'
        + eId + '<br>'
        + '<span style="width:100px">First Name:</span>'
        + efname + '<br>'
        + '<span style="width:100px">Last Name:</span>'
        + elname + '<br>'
        + '<span style="width:100px">Email:</span>'
        + eemail + '<br>'
        + '<span style="width:100px">Phone:</span>'
        + ephone + '<br>'
        + '<span style="width:100px">Department:</span>'
        + edepartment + '<br>'
    }
}
+ '<font color="green"><span align="left">'
+ efname + '</font></span><br>'
+ '<span style="width:100px">Last Name:</span>'
+ '<font color="green"><span align="left">'
+ elname + '</font></span><br>'
+ '<span style="width:100px">Email:</span>'
+ '<font color="green"><span align="left">'
+ eemail + '</font></span><br>'
+ '<span style="width:100px">Phone:</span>'
+ '<font color="green"><span align="left">'
+ ephone + '</font></span><br>'
+ '<span style="width:100px">Department:</span>'
+ '<font color="green"><span align="left">'
+ edepartment + '</font></span>';}
}

// Error handler for the asynchronous functions.
var myErrorHandler = function(statusCode, statusMsg)
{
 alert('Status: ' + statusCode + ', ' + statusMsg);
}

</script>
</head>

<body>
<!---- The form to display the employee drop-down list and
employee data. --->
<form name="simpleAJAX" method="get">
List of Employees:
<select name="employee" onChange="getEmployeeDetails(this.value)">
<script language="javascript">
getEmployees();
</script>
</select>
<br><br>
<span id="empData"></span>
</form>
The following component, which gets the data from the data source, must be in a file named emp.cfc in the components subdirectory of the application directory. The CFC uses the cfdocexamples data source that is installed with ColdFusion if you install the documentation.

```cfml
<cfc component>
  <cfset this.dsn = "cfdocexamples"/>
  <cffunction name="getEmployees" access="remote" returnFormat="json"
      output="false">
    <cfargument name="empid" required="no" type="string" default="0"/>
    <cfquery name="qryEmp" datasource="#{this.dsn}"
        select * from Employees
        <cfif empid neq 0>
            where Emp_ID = #{empid}"
        </cfif>
    </Cfquery>
    <cfreturn qryEmp>
  </cffunction>
</cfc component>
```
This tag references a registered custom Java applet. To register a Java applet, in the ColdFusion Administrator, select Extensions > Java Applets. Using this tag within a `cfform` tag is optional. If you use it within `cfform`, and the `method` attribute is defined in the Administrator, the return value is incorporated into the form.

**Category**

*Forms tags*

**Syntax**

```xml
<cfapplet
    appletSource = "applet name"
    name = "form variable name"
    align = "alignment option"
    height = "height in pixels"
    hSpace = "space on each side in pixels"
    notSupported = "message to display for non-Java browser"
    param_1 = "applet parameter name"
    param_2 = "applet parameter name"
    param_n = "applet parameter name"
    vSpace = "space above and below in pixels"
    width = "width in pixels">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` whose value is a structure. Specify the structure name in the `attributeCollection` and use the tag's attribute names as structure keys.

---

**See also**

`cfform`, `cfformgroup`, `cfformitem`, `cfgrid`, `cfinput`, `cfobject`, `cfselect`, `cfervlet`, `cfslider`, `cfte xtarea`, `cftree`

**History**

ColdFusion MX:

- Removed the requirement that you use this tag within a `cfform` tag.
- Changed the behavior when this tag is used within a `cfform` tag; if the `method` attribute is defined in the Administrator, the return value of the applet's method is incorporated into the form.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appletSource</td>
<td>Required</td>
<td></td>
<td>Name of registered applet</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td><strong>Required</strong></td>
<td><strong>Form variable name for applet.</strong></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>-----------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>align</strong></td>
<td><strong>Optional</strong></td>
<td><strong>Alignment:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Left</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Right</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Bottom</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Top</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TextTop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Middle</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AbsMiddle</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Baseline</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AbsBottom</td>
<td></td>
</tr>
<tr>
<td><strong>height</strong></td>
<td><strong>Optional</strong></td>
<td><strong>Height of applet, in pixels.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>hSpace</strong></td>
<td><strong>Optional</strong></td>
<td><strong>Space on left and right of applet, in pixels.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>notSupported</strong></td>
<td><strong>Optional</strong></td>
<td><strong>See description</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Text to display if a page that contains a Java applet-based <code>cfform</code> control is opened by a browser that does not support Java or has Java support disabled, for example: <code>notSupported = &quot;&lt;b&gt;Browser must support Java to view ColdFusion Java Applets&lt;/b&gt;&quot;</code> Default value: <code>&lt;b&gt;Browser must support Java to view ColdFusion Java Applets!&lt;/b&gt;</code></td>
<td></td>
</tr>
<tr>
<td><strong>param_n</strong></td>
<td><strong>Optional</strong></td>
<td><strong>Registered parameter for applet. Specify only to override values for applet in ColdFusion Administrator.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>vSpace</strong></td>
<td><strong>Optional</strong></td>
<td><strong>Space above and below applet, in pixels.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>width</strong></td>
<td><strong>Optional</strong></td>
<td><strong>Width of applet, in pixels.</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Usage**
You can specify the applet `method` attribute only in the Administrator, Java Applets view. For other attributes, you can accept the default values in the Administrator view, or specify values in this tag and override the defaults. If Java applet components are stored in a JAR file, enter the information in the J2EE Archives > ColdFusion Administrator. For more information, see Embedding Java applets in the Developing ColdFusion Applications.

**Example**

```html
<p>cfapplet lets you reference custom Java applets that have been registered using the ColdFusion Administrator.<br>
To register a Java applet, open the ColdFusion Administrator and click "Applets" link under "extensions" section.<br>
This example applet copies text that you type into a form. Type some text, and then click "copy" to see the copied text.<br>

```html
<cfapplet appletsource = "copytext" name = "copytext">
</cfapplet>
```html
```
**cfapplication**

**Description**

Defines the scope of a ColdFusion application; enables and disables storage of Client variables; specifies the Client variable storage mechanism; enables Session variables; and sets Application variable time-outs.

**Category**

**Application framework tags**

**Syntax**

```cfapplication
<cfapplication
datasource="data_source_name"
name = "application name"
applicationTimeout = #CreateTimeSpan(days, hours, minutes, seconds)#
clientManagement = "yes|no"
clientStorage = "data_source_name|Registry|Cookie"
loginStorage = "cookie|session"
googlemapkey = "map key"
scriptprotect = "none|all|list"
serversideformvalidation = "yes|no"
sessionmanagement = "yes|no"
sessiontimeout = #CreateTimeSpan(days, hours, minutes, seconds)#
setclientcookies = "yes|no"
setdomaincookies = "yes|no"
compileextforinclude = "comma_separated_list"
strictnumbervalidation = "true|false">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` whose value is a structure. Specify the structure name in the `attributeCollection` and use the tag's attribute names as structure keys.

See also

cfassociate, cferror, cflock, cfmessagebox; Application.CFC Reference; Designing and Optimizing a ColdFusion Application and Integrating JEE and Java Elements in CFML Applications in the Developing ColdFusion Applications

**History**

ColdFusion 11: Added attributes - compileextforinclude and strictnumbervalidation

ColdFusion 9: Added datasource, googlemapkey, and serversideformvalidation attributeColdFusion

ColdFusion 8: Added secureJSON and SecureJSONPrefix attributesColdFusion MX 7: Added scriptProtect attributeColdFusion MX 6.1: Added loginStorage attributeColdFusion MX:

- Changed how persistent scopes are available: Server, Session, and Application scope variables are stored in memory as structures. In earlier releases, only Session and Application scope variables were stored this way. You cannot access the UDF function scope as a structure.
- Changed the algorithm for setting the CFTOKEN variable value: if the registry key UUIDToken is a nonzero
value, ColdFusion uses a number constructed from the UUID plus a random number. Otherwise, ColdFusion sets the CFTOKEN variable default value using a positive random integer. (In earlier releases, ColdFusion always used a number constructed from the UUID plus a random number.)

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authCookie</td>
<td>Optional</td>
<td></td>
<td>Struct containing ColdFusion Authentication cookie related properties</td>
</tr>
<tr>
<td>datasource</td>
<td>Optional</td>
<td></td>
<td>Name of the data source from which the query retrieves data.</td>
</tr>
<tr>
<td>name</td>
<td>See Description</td>
<td></td>
<td>Name of application. Up to 64 characters. For Application and Session variables: Required. For Client variables: Optional</td>
</tr>
<tr>
<td>applicationTimeout</td>
<td>Optional</td>
<td>Specified in Variables page of ColdFusion Administrator</td>
<td>Lifespan of application variables. CreateTimeSpan function and values in days, hours, minutes, and seconds, separated by commas.</td>
</tr>
<tr>
<td>clientManagement</td>
<td>Optional</td>
<td>no</td>
<td>• yes: enables client variables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no</td>
</tr>
<tr>
<td>clientStorage</td>
<td>Optional</td>
<td>registry</td>
<td>How client variables are stored:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• datasource_name: in ODBC or native data source. Create storage repository in the Administrator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• registry: in the system registry.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• cookie: on client computer in a cookie. Scalable. If client disables cookies in the browser, client variables do not work.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>exchangeServerVersion</code></td>
<td>Optional</td>
<td>2007</td>
<td>Specifies the Microsoft Exchange Server version. The values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2007</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If you do not specify the details, 2007 is taken by default.</td>
</tr>
<tr>
<td><code>googleMapKey</code></td>
<td>Optional</td>
<td></td>
<td>The Google Maps API key required to embed Google Maps in your web pages.</td>
</tr>
<tr>
<td><code>loginStorage</code></td>
<td>Optional</td>
<td><code>cookie</code></td>
<td>• <code>cookie</code>: store login information in the Cookie scope.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <code>session</code>: store login information in the Session scope.</td>
</tr>
<tr>
<td><code>scriptProtect</code></td>
<td>Optional</td>
<td></td>
<td>Determined by ColdFusion Administrator Enable Global Script Protection setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specifies whether to protect variables from cross-site scripting attacks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <code>none</code>: do not protect variables</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <code>all</code>: protect Form, URL, CGI, and Cookie variables</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• comma-delimited list of ColdFusion scopes: protect variables in the specified scopes. For more information, see Usage.</td>
</tr>
<tr>
<td><code>secureJSON</code></td>
<td>Optional</td>
<td><code>Administrator value</code></td>
<td></td>
</tr>
<tr>
<td><code>serverSideFormValidation</code></td>
<td>Optional</td>
<td><code>yes</code></td>
<td>If no, disables validation on cfform fields when the form is submitted.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>secureJSONPrefix</td>
<td>Optional</td>
<td>Administrator value</td>
<td>The security prefix to put in front of the value that a ColdFusion function returns in JSON-format in response to a remote call if the secureJSON setting is true. The default value is the value of the Prefix serialized JSON setting in the Administrator Server Settings &gt; Settings page (which defaults to //, the JavaScript comment character). For more information see <a href="#">Improving security</a> in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>sessionCookie</td>
<td>Optional</td>
<td>Struct containing ColdFusion session cookie related properties.</td>
<td></td>
</tr>
</tbody>
</table>
| sessionManagement  | Optional| no                           | • yes: enables session variables.  
• no                                                                                                                                                     |
| sessionTimeout     | Optional| Specified in Variables page of ColdFusion Administrator | Life span of session variables. `CreateTimeSpan` function and values in days, hours, minutes, and seconds, separated by commas.               |
| setClientCookies   | Optional| yes                          | • yes: enables client cookies.  
• no: ColdFusion does not automatically send CFID and CFTOKEN cookies to client browser; you must manually code CFID and CFTOKEN on the URL for every page that uses Session or Client variables. |
<table>
<thead>
<tr>
<th>setDomainCookies</th>
<th>Optional</th>
<th>no</th>
</tr>
</thead>
</table>
| • **yes**: uses domain cookies for CFID and CFTOKEN cookies and for all Client variables when using cookies for client variable storage. Required for applications running on clusters.  
• **no**: uses host-specific cookies for CFID, CFTOKEN, and all client variable cookies. |

<table>
<thead>
<tr>
<th>compileextforinclude</th>
<th>Optional</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the list of allowed file extensions (as a comma separated list) for the <code>&lt;cfinclude&gt;</code> tag. Specifying a wildcard (<code>*</code>) in the list makes any file included using the <code>&lt;cfinclude&gt;</code> tag to be compiled. If any file included using the <code>&lt;cfinclude&gt;</code> tag is not found in this list, their content will be statically included. By default, files with the extensions cfm and cfml are always compiled irrespective of this setting. See <a href="https://help.adobe.com">cfinclude</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>strictnumbervalidation</td>
<td>Optional</td>
<td>true</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>------</td>
</tr>
</tbody>
</table>

In ColdFusion 10 and earlier versions, the `isValid` function allowed currency symbols at the start and commas inside the number.

Starting from ColdFusion 11, this function evaluates on a more strict basis. Setting this value to false makes the `isValid` function to behave in the older way. This setting effects `cfargument`, `cfparam` and `cfform` tags wherever integer & numeric validation is used. Based on this setting, the validation reflects in those tags as well.

### Usage

This tag is typically used in the Application.cfm file, to set defaults for a ColdFusion application.

⚠️ **Note**

You can also set the application defaults in the Application.cfc file. For more information, see Application variables.

This tag enables application variables, unless they are disabled in the ColdFusion Administrator. The Administrator setting also overrides the `sessionManagement` attribute. For more information, see Configuring and Administering ColdFusion. If ColdFusion is running on a cluster, specify `clientStorage = "cookie"` or a data source name; you cannot specify "registry". ColdFusion generates an error if the application name is longer than 64 characters. The CFTOKEN variable is 8 bytes in length. Its range is 10000000 -99999999.

⚠️ **Note**

If you specify `ClientStorage=cookie`, any Client scope variables set following a `cfflush` tag are not saved in the Client browser.

### Protecting variables from cross-site scripting attacks

The `ScriptProtect` attribute lets you protect one or more variable scopes from cross-site scripting attacks, where a client attempts to get your application to send malicious code back to a user's browser. In these attacks, user input (for example, from form fields or from URL variables) sets a CF variable which is destined for user output. The submitted data includes malicious code, such as JavaScript or an applet or object reference, which then executes on the user's system.
The ColdFusion Administrator Settings page Enable Global Script Protection option determines the default script protection setting. You can use the `scriptProtect` attribute to override the Administrator setting. You can also use the Application.cfc initialization code to set the protection value.

The ColdFusion cross-site scripting protection operation is done when ColdFusion processes the application settings at the beginning of a request. Thus, it can process the URL, and Cookie, CGI, and Form variables in a user's request. By default, it replaces occurrences of the following HTML tag names with the text `InvalidTag: object, embed, script, applet, and meta`. It allows these names in plain text, and replaces the words if they are used as tag names. You can specify any or all ColdFusion scopes for protection, but only the Form, URL, CGI, and Cookie scopes have variables that are often provided by unknown sources. Also, protecting a scope requires additional processing. For these reasons, the `all` attribute value applies protection to only the four scopes.

The script protection mechanism applies a regular expression that is defined in the `cf_root/lib/neo-security.xml` file in the server configuration, or the `cf_root/WEB-INF/cfusion/lib/neo-security.xml` file in the JEE configuration to the variable value. You can customize the patterns that ColdFusion replaces by modifying the regular expression in the `CrossSiteScriptPatterns` variable.

**Locking server, application, and session variables**

When you set or update variables in the server, application, and session scopes, use the `cflock` tag with the `scope` attribute set to the following value:

- For server variables, specify `server`
- For application variables, specify `application`
- For session variables, specify `session`

In some cases, you must also lock code that reads variables in these scopes. For information about locking scopes, see `cflock`.

**Example**

---

© 2014 Adobe Systems Incorporated. All rights reserved.
<h3>cfapplication Example</h3>
<p>cfapplication defines scoping for a ColdFusion application and enables or disables application and/or session variable storage. This tag is placed in a special file called Application.cfm that automatically runs before any other CF page in a directory (or subdirectory) where the Application.cfm file appears.</p>

```
<cfapplication name = "ETurtle"
  sessionTimeout = #CreateTimeSpan(0, 0, 0, 60)#
  sessionManagement = "Yes">

  <!--- Initialize session and application variables used by E-Turtleneck. --->
  <cfparam name="application.number" default="1">
  <cfparam name="session.color" default="">
  <cfparam name="session.size" default="">

  <!--- Use the application variable to prevent race condition. This variable keeps track of total number of turtlenecks sold. --->
  <cflock scope = "Application" timeout = "30" type = "Exclusive">
    <cfset application.number = application.number + session.numPurchased>
  </cflock>

  E-Turtleneck is proud to say that we have sold #application.number# turtlenecks to date.
```

<!---- End of Application.cfm --->
**cfargument**

**Description**

Creates a parameter definition within a component definition. Defines a function argument. Used within a **cffunction** tag.

**Category**

**Extensibility tags**

**Syntax**

```xml
<cfargument
    name="string"
    default="default value"
    displayname="descriptive name"
    hint="extended description"
    required="yes|no"
    type="data type">
```

**See also**

**cfcomponent, cffunction, cfinterface, cfinvoke, cfinvokeargument, cfobject, cfproperty, cfreturn**

**History**

ColdFusion 10: Added the following REST attributes: `restArgSource, restArgName`}

ColdFusion 8: Added `{component as a valid value for the ReturnType attribute.`

ColdFusion MX 7: Added the `xml` value of `type` attribute.

ColdFusion MX: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td>String; an argument name.</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>Optional</td>
<td></td>
<td>If no argument is passed, specifies a default argument value.</td>
</tr>
<tr>
<td>displayname</td>
<td>Optional</td>
<td>name attribute value</td>
<td>Meaningful only for CFC method parameters. A value to display when using introspection to show information about the CFC.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Optional</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>hint</td>
<td>Optional</td>
<td></td>
<td>Meaningful only for CFC method parameters. Text to display when using introspection to show information about the CFC. The <code>hint</code> attribute value follows the <code>displayname</code> attribute value in the parameter description line. Use this attribute to describe the purpose of the parameter.</td>
</tr>
<tr>
<td>required</td>
<td>Optional</td>
<td>no</td>
<td>Note: All arguments are required when invoked as a web service, irrespective of how they are defined. Specifies whether the parameter is required to execute the component method. The parameter is not required if you specify a <code>default</code> attribute.</td>
</tr>
<tr>
<td>restArgSource</td>
<td>Optional</td>
<td></td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <code>path</code>: Extracts parameters from the resource URL. URI path parameters are extracted from the request URI, and the parameter names correspond to the URI path template variable names specified in the <code>restPath</code> specified in <code>cffunction</code>. The default value cannot be used with path param.</td>
</tr>
</tbody>
</table>
• **query**: Extracts parameters from URL query parameters. Mostly used with HTTP GET method. When using GET method to send data, the parameters and values are encoded into the URL. QueryParam is used to extract these values and assign them to the appropriate variables.

• **form**: Extracts parameters from a form submission. It is used with HTTP POST method.

• **cookie**: Extracts values from a cookie.

• **header**: Extracts parameters from HTTP header.

• **matrix**: Extracts parameters from matrix URI. For details, see [http://www.w3.org/DesignIssues/MatrixURIs.html](http://www.w3.org/DesignIssues/MatrixURIs.html). If no value is specified, parameters are taken from the body of the request. Using `cfhttp`, you can send arguments in either form or body. For form parameter, the argument is consumed if you specify form as the value for `restArgSource` and for body parameter, if you do not specify `restArgSource`.

  While calling the service, if you pass the argument in body, in some scenarios, it is accepted as form. For example, `<cfhttpparam type="body" value="arg=someva...`
This is because, when you pass the form name-value pair through `cfhttp` or the body as "name=value",
the body content is same, due to which REST service is not able to detect what is passed.
In the service, if you expect body argument but while accessing the service through `cfhttp`, no argument in the body is passed, you will still receive the body argument as empty string.
If you use this attribute, but do not specify the
**restArgName**  
Optional  

The name that can be mapped to the argument name. While calling functions, arguments are extracted from the incoming request. If `restArgName` is provided, it is searched in the `restArgSource` scope of the request to populate the argument. If not specified, argument name is searched. If specified, then argument name has no impact. You can specify correct argument name so that it corresponds to the appropriate parameter. If the argument name passed to a REST service is not a valid Java identifier, for example, the name has hyphen (say `arg-name`), then you can use this attribute to map the argument name.

**type**  
Optional  

String: a type name; data type of the argument.
Usage

- any
- array
- binary
- boolean
- component: the argument must be a ColdFusion component.
- date
- guid: the argument must be a UUID or GUID of the form `xxxxxxx-xxxx-xxxx-xxxx-xxxxxxx` where each `x` is a character representing a hexadecimal number (0-9A-F).
- numeric
- query
- string
- struct
- uuid: the argument must be a ColdFusion UUID of the form `xxxxxxx-xxxx-xxxx-xxxx-xxxxxxx` where each `x` is a character representing a hexadecimal number (0-9A-F).
- variableName: a string formatted according to ColdFusion variable naming conventions.
- xml: XML objects and XML strings
- a component name: if the type attribute value is not one of the preceding items, ColdFusion treats it as the name of a ColdFusion component. When the function executes, it generates an error if the argument that is passed in is not a CFC with the specified name.
This tag must be in a `cffunction` tag, before any other tags in the `cffunction` tag body. Arguments that are passed when a method is invoked can be accessed from the method body in the following ways:

- With shorthand syntax: `#myargument#` (This example accesses the argument `myargument`.)
- Using the arguments scope as an array: `#arguments[1]#` (This example accesses the first defined argument in the `cffunction`.)
- Using the arguments scope as a struct: `#arguments.myargument#` (This example accesses the argument `myargument` in the array.)

Example

```coldfusion
<!--- This example defines a function that takes a course number parameter and returns the course description. --->
<cffunction name="getDescription">
  <!--- Identify argument. --->
  <cfargument name="Course_Number" type="numeric" required="true">
  <!--- Use the argument to get a course description from the database. --->
  <cfquery name="Description" datasource="cfdocexamples">
    SELECT Descript
    FROM Courses
    WHERE Number = '#Course_Number#'
  </cfquery>
  <!--- Specify the variable that the function returns. --->
  <cfreturn Description.Descript>
</cffunction>
```
cfassociate

Description

Allows subtag data to be saved with a base tag. Applies only to custom tags.

Category

Application framework tags

Syntax

```
<cfassociate
   baseTag = "base tag name"
   dataCollection = "collection name">
```

Note

You can specify the attributes of this tag in an attributeCollection whose value is a structure. Specify the structure name in the attributeCollection and use the tag’s attribute names as structure keys.

See also

cfapplication, cferror, cflock, cfmessagebox; High-level data exchange in Nesting custom tags in the Developing ColdFusion Applications.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseTag</td>
<td>Required</td>
<td></td>
<td>Base tag name.</td>
</tr>
<tr>
<td>dataCollection</td>
<td>Optional</td>
<td>AssocAttribs</td>
<td>Structure in which base tag stores subtag data.</td>
</tr>
</tbody>
</table>

Usage

Call this tag within a subtag, to save subtag data in the base tag.
When ColdFusion passes subtag attributes back to the base tag, it saves them in a structure whose default name is AssocAttribs. To segregate subtag attributes (in a base tag that can have multiple subtags), specify a structure name in the dataCollection attribute. The structure is appended to an array whose name is thistag.collectionName.
In the custom tag code, the attributes passed to the tag by using the cfmodule tag attributeCollection attribute are saved as independent values, with no indication that they are grouped into a structure by the custom tag’s caller. Therefore, in the called tag, if you assign a value to a specific attribute, it replaces the value passed in the attributeCollection attribute that you used when calling the subtag.

Example
<!--- Find the context. --->
<cfif thisTag.executionMode is "start">
<!--- Associate attributes. --->
<cfassociate baseTag = "CF_TAGBASE">

<!--- Define defaults for attributes. --->
<cfparam name = "attributes.happy" default = "yes">
<cfparam name = "attributes.sad" default = "no">
...

cfauthenticate

Description

This tag is obsolete. Use the newer security tools; see Conversion functions and Securing Applications in the Developing ColdFusion Applications.

History

ColdFusion MX: This tag is obsolete. It does not work in ColdFusion MX and later releases.
**cfbreak**

**Description**

Used within a `cfloop` tag. Breaks out of a loop.

**Category**

*Flow-control tags*

**Syntax**

```
<cfbreak>
```

**See also**

*cfabort, cfcontinue, cfexecute, cfif, cflocation, cfloop, cfthrow, cftry; cfloop and cfbreak* in the *Developing ColdFusion Applications*

**Tag example**

```
<cfloop index="i" from="1" to="10">
  <cfoutput>#i#</cfoutput><br>
  <cfif i Ge 5>
    <cfbreak>
  </cfif>
</cfloop>
```

**Script example**

```
for (i=1; i <= 10; i++){
    writeOutput("#i#<br>");
    if (i >= 5){
        break;
    }
}
```

Either version of the code above outputs:

1
2
3
4
5
Tags c

This section lists tags starting with alphabet C with their description.

- cfcache
- cfcalendar
- cfcase
- cfcatch
- cfchart
- cfchartdata
- cfchartseries
- cfcol
- cfcollection
- cfcomponent
- cfcontent
- cfcontinue
- cfcookie
- cfclient
- cfclientsettings
cfcache

Description

Stores a copy of a page on the server and/or client computer, to improve page rendering performance. To do this, the tag creates temporary files that contain the static HTML returned from a ColdFusion page. Use this tag if it is not necessary to get dynamic content each time a user accesses a page. You can use this tag for simple URLs and for URLs that contain URL parameters.

Category

Page processing tags

Syntax

```xml
<cfcache
    action = "action"
    dependsOn = "variable name list"
    directory = "directory path"
    expireURL = "wildcarded URL reference"
    id = "object identifier"
    idleTime = "decimal number of days"
    metadata = "variable name"
    name = "variable name"
    password = "password"
    port = "port number"
    protocol = "http://|https://"
    region = "region_name"
    stripWhiteSpace = "false|true"
    throwOnError = "false|true"
    timespan = "decimal number of days">
    useCache = "true|false"
    usequerystring = "false|true"
    username = "username"
    value = "value">
    The page fragment to be cached, if any.
</cfcache>
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfflush, cfheader, cfhtmlhead, cfsetting, cfsilent, Cache_functions

History

ColdFusion 10: Added the attribute {{region}}

ColdFusion 9:

• Added support for the following features:
- Caching in memory. Memory is now the default cache location.
- Caching page fragments.
- Caching specific objects, including the ability to put, get, and flush cached objects.
- Setting cache dependencies.
- Setting an idle timeout.
- Getting metadata about cached objects.
- The ability to strip white space from cached page fragments.
- The ability to throw an exception if an error occurs when flushing a cached object.

- **Added** `get` and `put` values of the `action` attribute. These values support caching of objects.
- **Added** `dependsOn`, `id`, `idleTime`, `key`, `metadata`, `name`, `stripWhiteSpace`, `throwOnError`, `useCache`, `useQueryString`, and `value` attributes.

ColdFusion MX:
- Deprecated the `cachedirectory` and `timeout` attributes. They might not work, and might cause an error, in later releases.
- **Added** the `timespan` attribute.
- Changed how pages are cached: the default `action` attribute value, `cache`, caches a page on the server and the client. (In earlier releases, this option cached a page only on the server.)
- Changed the source of the `protocol` and `port` values: the default `protocol` and `port` values are now taken from the current page URL. (In earlier releases, they were "http" and "80", respectively.)
- Changed how session state is handled when caching a page: this tag can cache pages that depend on session state, including pages that are secured with a ColdFusion login. (In earlier releases, the session state was cleared when caching the page, causing authentication to be lost.)
- Changed how files are cached: this tag uses a hash() of the URL for the filename to cache a file. (In earlier releases, ColdFusion used the `cfcache.map` file.)

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Actions</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### action

<table>
<thead>
<tr>
<th>All</th>
<th>Optional</th>
<th>serverCache</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>server-side and client-side page caching.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>clientcache: browser-side page caching only. To cache a personalized page, use this option.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>flush: remove the current versions of cached pages, fragments, or an object from the cache. The cache is refreshed the next time a user accesses the item. For more information see Description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>get: get an object from the cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optimal: same as cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>put: Add an object to the cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>serverCache: server-side caching only.</td>
</tr>
</tbody>
</table>

### dependsOn

| cache, serverCache, optimal | Optional | A comma separated list of variables. If any of the variable values change, ColdFusion updates the cache. This attribute can take an expression that returns a list of variables. |

### directory

<p>| cache, serverCache, clientCache, optimal, flush, put | Optional | Cache in memory | Absolute path of cache directory. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Parameters</th>
<th>Default Compatibility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expireURL</td>
<td>flush</td>
<td>Optional</td>
<td>A URL reference. Can include wildcards, for example: &quot;*/view.cfm?id=&quot;. ColdFusion flushes, from the cache, pages that match the specified URL or pattern.</td>
</tr>
<tr>
<td>id</td>
<td>flush, get, put</td>
<td>See description</td>
<td>The identifier for a cached object. This attribute can also take a comma-separated list of identifiers. This attribute is required for the any operation on an object. Therefore it is required for the get and put actions, and to flush an object. It is not required to flush a page.</td>
</tr>
<tr>
<td>idleTime</td>
<td>cache, serverCache, clientCache, optimal, flush, put</td>
<td>Optional</td>
<td>No idle timeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Flushes the cached item if it is not accessed for the specified time span:</td>
</tr>
</tbody>
</table>
|             |                     |                       | • A decimal number of days, for example: ".25", for one-fourth day (6 hours); "1", for one day; "1.5", for one and one half days  
<p>|             |                     |                       | • A return value from the <a href="#">CreateTimeSpan</a> function, for example, &quot;#{CreateTimeSpan(0, 6, 0, 0)}&quot;.                                                                                                                                                                       |</p>
<table>
<thead>
<tr>
<th>metadata</th>
<th>get</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>The name of a structure variable in which to put object metadata. The get operation returns the following data:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• timespan: The time span during which the cached item is valid; the value of the timespan attribute for the cached item.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• createdtime: The time when the cache was created</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• lasthit: The time when the cached item was most recently used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• hitnumber: Number of times the cached item has been used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• missnumber: Number of misses</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>get</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>The name of the variable in which to put the retrieved object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>password</th>
<th>cache, serverCache, clientCache, optimal, flush</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A password. Provide this if the page requires authentication at the web-server level.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>port</td>
<td>The port for the current page. Port number of the web server from which the URL for the cached page is requested. In the internal call from <code>cfcache</code> to <code>cfhttp</code>, ColdFusion resolves each URL variable in the page; this ensures that links in the page remain functional.</td>
<td>Optional</td>
</tr>
<tr>
<td>protocol</td>
<td>The current page protocol. Protocol that is used to create URL from cache.</td>
<td>Optional</td>
</tr>
<tr>
<td>region</td>
<td>Name that you assign to a cache region.</td>
<td>Optional</td>
</tr>
<tr>
<td>stripWhiteSpace</td>
<td>Specifies whether to strip any unnecessary white space characters from a cached page fragment. Does not have any effect on cached pages or objects.</td>
<td>Optional</td>
</tr>
<tr>
<td>throwOnError</td>
<td>A Boolean value specifying whether to throw an error if the <code>flush</code> action encounters an error. Otherwise the action does not generate an error if it fails. If this attribute is <code>true</code> you can handle the error in a <code>cfcatch</code> block, for example, if a specified id value is invalid.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional</td>
</tr>
</tbody>
</table>
| timespan            | cache, serverCache, clientCache, optimal, flush, put | Optional | See Description | The interval until the item is flushed from the cache.  
|---------------------|------------------------------------------------------|----------|-----------------|---------------------------------------------------------------------  
|                     |                                                      |          |                 | • A decimal number of days, for example: ".25", for one-fourth day (6 hours); "1", for one day; "1.5", for one and one half days  
|                     |                                                      |          |                 | • A return value from the `CreateTimeSpan` function, for example, "#CreateTimeSpan(0, 6, 0, 0)". The default action is to flush the item when it is idle for the time specified by the `idleTime` attribute, or `cfcache action = "flush"` executes.  
<p>| useCache            | cache, serverCache, optimal,                        | Optional | true            | Specifies whether to use caching for a page. This attribute can be useful during development. For example, you could use a function to predict when to use a cache, based on the application state. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usequerystring</td>
<td>Optional</td>
<td>false</td>
<td>If true, generates a template cache ID that includes the query string. This means that a new template cache is created whenever the query string changes. If set to true, the attribute depends on considers the URL parameters defined in the query string as well to generate template caches. Also see the Usage section.</td>
</tr>
<tr>
<td>username</td>
<td>cache, serverCache, clientCache, optimal, flush</td>
<td>Optional</td>
<td>A username. Provide this if the page being cached or flushed requires authentication at the web server level.</td>
</tr>
<tr>
<td>value</td>
<td>put</td>
<td>Required</td>
<td>The object to cache.</td>
</tr>
</tbody>
</table>

**Usage**

**Page fragments:** To cache a page fragment, put the fragment in the body of the tag, between the begin tag and the end tag. Do not use a tag body to cache full pages or objects.

**flush:** The flush action can have two formats: One uses the `ExpiresURL` attribute to specify the page to flush, the other uses the `id` attribute to specify the object to flush. When you flush an object, ColdFusion ignores errors by default. If you specify a `throwOnError` attribute with a `true` value, the action throws the errors, and you can use a catch block to handle them. This is useful to determine if you use invalid cache ID values.

**User-defined cache:** To create user-defined cache,

1. Add the following snippet to the ehcache.xml (in the _CF_root_lib):
1. To configure the cache, use the following configuration:

```xml
<cache name="cf9"
    maxElementsInMemory="10000"
    eternal="false"
    timeToIdleSeconds="86400"
    timeToLiveSeconds="86400"
    overflowToDisk="true"
    diskSpoolBufferSizeMB="30"
    maxElementsOnDisk="10000000"
    diskPersistent="true"
    diskExpiryThreadIntervalSeconds="3600"
    memoryStoreEvictionPolicy="LRU"/>
```

2. To reference the user-defined cache, use the key attribute as follows:

```cfml
<cfcache key="cf9" timespan=#createtimespan(0,0,1,0)# >
    <cfoutput>$now()$</cfoutput>
</cfcache>
```

By default, caching is memory-based and not disk-based. For each application, the default setting is 10000 object caches and 10000 template caches. It is important to note the limit imposed on the number of objects/templates that can be cached.

Disk overflow for caching by default is false. To enable disk caching, set `overflowToDisk` as true in the `ehcache.xml`. To make the cached data available on server restart, set `diskPersistent` to true.

For further details of the properties in the `ehcache.xml`, refer to the documentation available at the following URL: [http://ehcache.org/](http://ehcache.org/)

**From ColdFusion 8 and earlier**

The following remaining information for this tag also applied to previous releases.

Use this tag in pages whose content is not updated frequently. Taking this action can greatly improve the performance of your application.

The output of a cached page is stored in a file on the client browser and/or the ColdFusion server. Instead of regenerating and downloading the output of the page each time it is requested, ColdFusion uses the cached output. ColdFusion regenerates and downloads the page only when the cache is flushed, as specified by the `timespan` attribute, or by invoking `cfcache action=flush`.

To enable a simple form of caching, put a `cfcache` tag, specifying the `timespan` attribute, at the top of a page. Each time the specified time span passes, ColdFusion flushes (deletes) the copy of the page from the cache and caches a new copy for users to access.

You can specify client-side caching or a combination of client-side and server-side caching (the default), using the action attribute. The advantage of client-side caching is that it requires no ColdFusion resources; the browser stores pages in its own cache, improving performance. The advantage of combination caching is that it optimizes server performance; if the browser does not have a cache of the page, the server can get data from its own cache. (Adobe recommends that you use combination caching, and do not use server-side only caching.)

If a page contains personalized content, use the `action = "clientcache"` option to avoid the possibility of caching a personalized copy of a page for other users.

Debug settings have no effect on `cfcache` unless the application page enables debugging. When generating a cached file, `cfcache` uses `cfsetting showDebugOutput = "no"`.

The `cfcache` tag evaluates each unique URL, including URL parameters, as a distinct page, for caching purposes. For example, the output of [http://server/view.cfm?id=1](http://server/view.cfm?id=1) and the output of [http://server/view.cfm?id=](http://server/view.cfm?id=)
The `cfhttp` tag uses the `cfhttp` tag to get the contents of a page to cache. If there is an HTTP error accessing the page, the contents are not cached. If a ColdFusion error occurs, the error is cached.

For more information, see Caching ColdFusion pages that change infrequently in Optimizing ColdFusion applications in the Developing ColdFusion Applications.

### Change in behaviour

Till ColdFusion 9, the request query string was automatically used as part of the page identifier, so that pages with different query strings (URL parameters) are cached independent of each other. However, you must now specify the new optional attribute `usequerystring` with value set to `true`, to achieve the older behavior.

#### Example

```coldfusion
<!---- This example produces as many cached files as there are URL parameter permutations. You can see that the page is cached when the timestamp doesn't change.---->

<cfhttp
    timespan="#createTimeSpan(0,0,10,0)#">
<body>
<h3>This is a test of some simple output</h3>
<cfoutput>
    This page was generated at #now()#<br>
</cfoutput>
<cfparam name = "URL.x" default = "no URL parm passed">
<cfoutput>
    The value of URL.x = # URL.x #</cfoutput>
```
**cfcalendar**

**Description**

Puts an interactive Flash format calendar in an HTML or Flash form. Not supported in XML format forms. The calendar lets a user select a date for submission as a form variable.

**Category**

**Forms tags**

**Syntax**

```xml
<cfcalendar
   name = "name of calendar"
   dayNames = "days of the week labels"
   disabled = "yes|no|no attribute value"
   enabled = "yes|no"
   endRange = "last disabled date"
   height = "height"
   mask = "character pattern"
   monthNames = "month labels"
   onBlur = "ActionScript to invoke"
   onChange = "ActionScript to invoke"
   onFocus = "ActionScript to invoke"
   selectedDate = "date"
   startRange = "first disabled date"
   style="Flash ActionScript style"
   tooltip = "text"
   visible = "yes|no"
   width = "width">
</cfcalendar>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfform, cfgrid, cfinput, cfselect, cfslider, cftextarea, cftree; About Flash form styles in the Developing ColdFusion Applications.

**History**

ColdFusion MX 7.01: Added support for `onBlur` and `onFocus` events. ColdFusion MX 7: Added tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>The name of the calendar.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Optional</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dayNames</td>
<td>Optional</td>
<td>S, M, T, W, Th, F, S</td>
<td>A comma-delimited list that sets the names of the weekdays displayed in the calendar. Sunday is the first day and the rest of the weekday names follow in the normal order.</td>
</tr>
<tr>
<td>disabled</td>
<td>Optional</td>
<td>Not disabled</td>
<td>Disables all user input, making the control read-only. To disable input, specify <code>disabled</code> without an attribute or <code>disabled=&quot;Yes&quot;</code> (or any ColdFusion positive boolean value, such as <code>true</code>). To enable input, omit the attribute or specify <code>disabled=&quot;No&quot;</code> (or any ColdFusion negative Boolean value, such as <code>false</code>).</td>
</tr>
<tr>
<td>enabled</td>
<td>Optional</td>
<td>yes</td>
<td>Flash only: Boolean value that specifies whether the control is enabled. A disabled control appears in light gray. This is the inverse of the <code>disabled</code> attribute.</td>
</tr>
<tr>
<td>endRange</td>
<td>Optional</td>
<td></td>
<td>The end of a range of dates that are disabled. Users cannot select dates from the date specified by the <code>startRange</code> attribute through this date.</td>
</tr>
<tr>
<td>firstDayOfWeek</td>
<td>Optional</td>
<td>0</td>
<td>Integer in the range 0-6 specifying the first day of the week in the calendar: 0 indicates Sunday; 6 indicates Saturday.</td>
</tr>
<tr>
<td>height</td>
<td>Optional</td>
<td>Determined by Flash</td>
<td>The vertical dimension of the calendar specified in pixels.</td>
</tr>
</tbody>
</table>
| **mask** | **Optional** | **MM/DD/YYYY** | A pattern that specifies the format of the submitted date. Mask characters are:  
- **D** = day; can use 0-2 mask characters  
- **M** = month; can use 0-4 mask characters  
- **Y** = year; can use 0, 2, or 4 characters  
- **E** = day in week; can use 0-4 characters  
- Any other character = put the character in the specified location  
For more information on masking, see “Masking input data” in the `cfin` reference page. |
<p>| <strong>monthNames</strong> | <strong>Optional</strong> | <strong>January, February, March, April, May, June, July, August, September, October, November, December</strong> | A comma-delimited list of the month names that are displayed at the top of the calendar. |
| <strong>onBlur</strong> | <strong>Optional</strong> |  | ActionScript that runs when the calendar loses focus. |
| <strong>onChange</strong> | <strong>Optional</strong> |  | ActionScript that runs when the user selects a date. |
| <strong>onFocus</strong> | <strong>Optional</strong> |  | ActionScript that runs when the calendar gets focus. |
| <strong>selectedDate</strong> | <strong>Optional</strong> | <strong>None (Flash shows the current month)</strong> | The date that is initially selected. It is highlighted in a color determined by the form skin. Must be in <strong>mm/dd/yyyy</strong> or <strong>dd/mm/yyyy</strong> format, depending on the current locale. (Use the <code>setlocale</code> function to set the locale, if necessary.) |</p>
<table>
<thead>
<tr>
<th>startRange</th>
<th>Optional</th>
<th>The start of a range of dates that are disabled. Users cannot select dates from this date through the date specified by the endRange attribute.</th>
</tr>
</thead>
<tbody>
<tr>
<td>style</td>
<td>Optional</td>
<td>Flash ActionScript style or styles to apply to the calendar. For more information, see Setting styles and skins in Flash forms in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>tooltip</td>
<td>Optional</td>
<td>Flash only: Text to display when the mouse pointer hovers over the control.</td>
</tr>
<tr>
<td>visible</td>
<td>Optional</td>
<td>Flash only: Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank.</td>
</tr>
<tr>
<td>width</td>
<td>Optional</td>
<td>The horizontal dimension of the calendar specified in pixels.</td>
</tr>
</tbody>
</table>

**Usage**

The `cfcalendar` tag displays a calendar month, showing the month, the year, a grid of the days of the month, and headings for the days of the week. The calendar contains forward and back arrow buttons to let you change the month and year that are displayed.

If you include a value for the `selectedDate` attribute, that date is highlighted in green and determines the month and year that display initially. Changing the month and year display does not change the selected date. A user can change the selected date by clicking a different date on the calendar. The `onChange` attribute can specify an ActionScript event handler function that runs when the user selects a date.

The current date is highlighted in reverse (that is, a white number on a black background). If the selected date is in a different month or year, however, the current date does not appear unless you move to it by clicking the forward or back arrows.

The `mask` attribute lets you specify the format of the selected date that is returned to the application. You can use the keyboard to access and select dates from a `cfcalendar` control:

- Use the Up, Down, Left, and Right Arrow keys to change the selected date.
- Use the Home and End keys to reach the first and last enabled date in a month, respectively.
- Use the Page Up and Page Down keys to reach the previous and next month, respectively.
Note
The cfcalendar tag is not supported in XML format forms.

Example
This example produces a 200-pixel by 150-pixel calendar with a Flash haloBlue skin. It displays abbreviated month names and two-character days of the week. It initially displays today’s date as determined by the selectedDate attribute. When you click the Save button, the form submits back to the current page, which displays the submitted information.

The example also has three dateField controls that let the user change the initial selected date that displays on the calendar and a blocked-out date range. The initial blocked-out date is a four-day period immediately preceding today’s date.

Note
This example must be modified to work in locales that do not use mm/dd/yyyy date formats. To do so, use the LSDateFormat function in place of the DateFormat function and a mask that is appropriate for your locale, such as dd/mm/yyyy.

```cfm
<!--- Set initial selected and blocked-out dates.--->
<cfparam name="Form.startdate" default="#dateformat(now()-5, 'mm/dd/yyyy')#">
<cfparam name="Form.enddate" default="#dateformat(now()-1, 'mm/dd/yyyy')#">
<cfparam name="Form.selectdate" default="#dateformat(now(), 'mm/dd/yyyy')#">

<!--- If the form has been submitted, display the selected date. --->
<cfif isDefined("Form.submitit")>
  <cfoutput><b>You selected #Form.selectedDate#</b><br><br></cfoutput>
</cfif>

Please select a date on the calendar and click Save.</b><br><br>
<cfcalendar name="form1" format="Flash" skin="haloBlue" width="375" height="350" >
  <cfparam name="selectedDate" value="#Form.selectdate#">
  <cfparam name="startRange" value="#Form.startdate#">
  <cfparam name="endRange" value="#Form.enddate#">
  <cfparam name="mask" value="mmm dd, yyyy">
  <cfparam name="dayNames" value="SU, MO, TU, WE, TH, FR, SA">
  <cfparam name="firstDayOfWeek" value="first day of the week in integer">
  <cfparam name="monthNames" value="JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC">
  <cfparam name="style" value="rollOverColor: #FF0000">
  <cfparam name="width" value="200">
  <cfparam name="height" value="150">
  <cfinput type="dateField" name="startdate" label="Block out starts" width="100" value="#Form.startdate#">
  <cfinput type="dateField" name="enddate" label="Block out ends" width="100" value="#Form.enddate#">
  <cfinput type="dateField" name="selectdate" label="Initial date" width="100" value="#Form.selectdate#">
  <cfinput type="Submit" name="submitit" value="Save" width="100">
</cfcalendar>
```
**cfcase**

**Description**

Used only inside the `cfswitch` tag body. Contains code to execute when the expression specified in the `cfswitch` tag has one or more specific values.

**Category**

*Flow-control tags*

**Syntax**

```xml
<cfcase
    value = "value|delimited set of values"
    delimiters = "delimiter characters">
```

**See also**

`cfdefaultcase`, `cfswitch`, `cfcase`, and `cfdefaultcase` in the *Developing ColdFusion Applications*

**History**

ColdFusion 8: Changed the way ColdFusion parses `cfcase` values. Previously, `cfcase` tags with numeric value dates did not return expected results. For example, `\<cfcase value="00"\>` and `\<cfcase value="0A"\>` were both evaluated to 0. The value "0A" was treated as a date and converted to 0 number of days from 12/30/1899. The value "00" was also evaluated to the value 0. This caused the exception "Context validation error for tag CFCASE. The CFSWITCH has a duplicate CFCASE for value "0.0"." The `cfswitch` tag now returns the expected result.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Required</td>
<td></td>
<td>The value or values that the expression attribute of the <code>cfswitch</code> tag must match. To specify multiple matching values, separate the values with the <code>delimiters</code> character. The value or values must be simple constants or constant expressions, not variables.</td>
</tr>
<tr>
<td>delimiters</td>
<td>Optional</td>
<td>, (comma)</td>
<td>Specifies the delimiter character or characters that separate multiple values to match. If you specify multiple delimiter characters, you can use any of them to separate the values to be matched.</td>
</tr>
</tbody>
</table>

Usage

The contents of the cfcase tag body executes only if the expression attribute of the cfswitch tag evaluates to a value specified by the value attribute. The contents of the cfcase tag body can include HTML and text, and CFML tags, functions, variables, and expressions. You do not have to explicitly break out of the cfcase tag, as you do in some languages. One cfcase tag can match multiple expression values. To do this, separate the matching values with the default value of the delimiter character. For example the following line matches "red", "blue", or "green":

```html
<cfcase value="red,blue,green">
```

You can use the delimiter attribute to specify one or more delimiters to use in place of the comma. For example, the following line matches "cargo, live", "cargo, liquid", and "cargo, solid":

```html
<cfcase value="cargo, live;cargo, liquid-cargo, solid" delimiters=";,-">
```

Example

The following example displays a grade based on a 1-10 score. Several of the cfcase tags match more than one score. For simplicity, the example sets the score to 7.
<cfset score="7">
<cfswitch expression="#score#">
  <cfcase value="10">
    <cfset grade="A">
  </cfcase>
  <cfcase value="9;8" delimiters=";">
    <cfset grade="B">
  </cfcase>
  <cfcase value="7;6" delimiters=";">
    <cfset grade="C">
  </cfcase>
  <cfcase value="5;4;" delimiters=";">
    <cfset grade="D">
  </cfcase>
  <cfdefaultcase>
    <cfset grade="F">
  </cfdefaultcase>
</cfswitch>
<cfoutput>
  Your grade is #grade#
</cfoutput>
**cfcatch**

**Description**

Used inside a *cftry* tag. Together, they catch and process exceptions in ColdFusion pages. *Exceptions* are events that disrupt the normal flow of instructions in a ColdFusion page, such as failed database operations, missing include files, and developer-specified events.

**Category**

**Exception handling tags**

**Syntax**

```
<cfcatch type = "exception type">
  Exception processing code here </cfcatch>
```

**See also**

`cftry`, `cferror`, `cffinally`, `cfrethrow`, `cfthrow`, `onError`; *Handling Errors* in the *Developing ColdFusion Applications*

**History**

ColdFusion 10: Added the `sessionCookie` and `authCookie` attributes.

ColdFusion MX:

- Changed SQLSTATE value behavior: the SQLSTATE return value in a *cfcatch* tag depends on the database driver type:
  - Type 1 (JDBC-ODBC bridge): the value is the same as in ColdFusion 5.
  - Type 4 (100% Java, no native methods): the value might be different.
    If your application depends on SQLSTATE values for flow control, the application might produce unexpected behavior with ColdFusion MX.
- Changed the behavior of this tag when type="any": it is not necessary, when you include a *cfcatch* tag with type="any", to do so in the last *cfcatch* tag in the block, to ensure that all other tests are executed before it. ColdFusion finds the best-match *cfcatch* block.
- Changed the behavior of the *cfscript* tag: it includes *try* and *catch* statements that are equivalent to the *cftry* and *cfcatch* tags.
- Changed object modification: you cannot modify the object returned by *cfcatch*.
- Changed thrown exceptions: the *cfcollection*, *cfindex*, and *cfsearch* tags can throw the SEARCHENGINE exception. In earlier releases, an error in processing these tags threw only an UNKNOWN exception.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Optional</td>
<td></td>
<td>Variable name for cfcatch expression.</td>
</tr>
</tbody>
</table>
type | Optional | Any | *application*: catches application exceptions
*database*: catches database exceptions
*template*: catches ColdFusion page exceptions
*security*: catches security exceptions
*object*: catches object exceptions
*missingInclude*: catches missing include file exceptions
*expression*: catches expression exceptions
*lock*: catches lock exceptions
*custom_type*: catches the specified custom exception type that is defined in a `cfthrow` tag
*searchengine*: catches Solr search engine exceptions
*any*: catches all exception types

**Usage**

You must code at least one `cfcatch` tag within a `cftry` block. Put `cfcatch` tags at the end of a `cftry` block. ColdFusion tests `cfcatch` tags in the order in which they appear. This tag requires an end tag.

If `type="any"`, ColdFusion catches exceptions from any CFML tag, data source, or external object. To get the exception type use code such as the following:

```cfml
#cfcatch.type#
```

Applications can use the `cfthrow` tag to throw developer-defined exceptions. Catch these exceptions with any of these `type` options:

- "custom_type"
- "Application"
- "Any"

The `custom_type` type is a developer-defined type specified in a `cfthrow` tag. If you define a custom type as a series of strings concatenated by periods (for example, "MyApp.BusinessRuleException.InvalidAccount"), ColdFusion can catch the custom type by its character pattern. ColdFusion searches for a `cfcatch` tag in the `cftry` block with a matching exception type, starting with the most specific (the entire string), and ending with the least specific. For example, you could define a type as follows:
<cftrow type = "MyApp.BusinessRuleException.InvalidAccount">

If you have the following <cfcatch> tag, it handles the exception:

<cfcatch type = "MyApp.BusinessRuleException.InvalidAccount">

Otherwise, if you have the following <cfcatch> tag, it handles the exception:

<cfcatch type = "MyApp.BusinessRuleException">

Finally, if you have the following <cfcatch> tag, it handles the exception:

<cfcatch type = "MyApp">

You can code <cfcatch> tags in any order to catch a custom exception type. If you specify type = "Application", the <cfcatch> tag catches only custom exceptions that have the Application type in the cftrow tag that defines them. The cfindlude, cfmodule, and cferror tags throw an exception of type = "template". An exception that is thrown within a <cfcatch> block cannot be handled by the cftry block that immediately encloses the <cfcatch> tag. However, you can rethrow the currently active exception with the cfrethrow tag. The <cfcatch> variables provide the following exception information:

<table>
<thead>
<tr>
<th>cfcatch variable</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfcatch.type</td>
<td>Type: Exception type, as specified in cfcatch.</td>
</tr>
<tr>
<td>cfcatch.message</td>
<td>Message: Exception's diagnostic message, if provided; otherwise, an empty string; in the cfcatch.message variable.</td>
</tr>
<tr>
<td>cfcatch.detail</td>
<td>Detailed message from the CFML interpreter or specified in a cftrow tag. When the exception is generated by ColdFusion (and not cftrow), the message can contain HTML formatting and can help determine which tag threw the exception.</td>
</tr>
<tr>
<td>cfcatch.tagcontext</td>
<td>An array of tag context structures, each representing one level of the active tag context at the time of the exception.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cfcatch.NativeErrorCode</td>
<td>Applies to type = &quot;database&quot;. Native error code associated with exception. Database drivers typically provide error codes to diagnose failing database operations. Default value is -1.</td>
</tr>
<tr>
<td>cfcatch.SQLState</td>
<td>Applies to type = &quot;database&quot;. SQLState associated with exception. Database drivers typically provide error codes to help diagnose failing database operations. Default value is 1.</td>
</tr>
<tr>
<td>cfcatch.Sql</td>
<td>Applies to type = &quot;database&quot;. The SQL statement sent to the data source.</td>
</tr>
<tr>
<td>cfcatch.queryError</td>
<td>Applies to type = &quot;database&quot;. The error message as reported by the database driver.</td>
</tr>
<tr>
<td>cfcatch.where</td>
<td>Applies to type = &quot;database&quot;. If the query uses the cfqueryparam tag, query parameter name-value pairs.</td>
</tr>
<tr>
<td>cfcatch.ErrNumber</td>
<td>Applies to type = &quot;expression&quot;. Internal expression error number.</td>
</tr>
<tr>
<td>cfcatch.MissingFileName</td>
<td>Applies to type = &quot;missingInclude&quot;. Name of file that could not be included.</td>
</tr>
<tr>
<td>cfcatch.LockName</td>
<td>Applies to type = &quot;lock&quot;. Name of affected lock (if the lock is unnamed, the value is &quot;anonymous&quot;).</td>
</tr>
<tr>
<td>cfcatch.LockOperation</td>
<td>Applies to type = &quot;lock&quot;. Operation that failed (Timeout, Create Mutex, or Unknown).</td>
</tr>
<tr>
<td>cfcatch.ErrorCode</td>
<td>Applies to type = &quot;custom&quot;. String error code.</td>
</tr>
<tr>
<td>cfcatch.ExtendedInfo</td>
<td>Applies to type = &quot;application&quot; and &quot;custom&quot;. Custom error message; information that the default exception handler does not display.</td>
</tr>
</tbody>
</table>
<!--- The cfcatch example that uses TagContext to display the tag stack. --->
<h3>cftry Example</h3>
<!--- Open a cftry block. --->
<cftry>
<!--- Notice misspelled tablename "employees" as "employeas". --->
<cfquery name = "TestQuery" dataSource = "cfdocexamples">
  SELECT *
  FROM employees
</cfquery>
<!--- Other processing goes here. --->
<!--- Specify the type of error for which we search. --->
<cfcatch type = "Database">
<!--- The message to display. --->
<h3>You've Thrown a Database <b>Error</b></h3>
<cfoutput>
<!--- The diagnostic message from ColdFusion. --->
  <p>#cfcatch.message#</p>
  <p>Caught an exception, type = #CFCATCH.TYPE#</p>
  <p>The contents of the tag stack are:</p>
  <cfdump var="#cfcatch.tagcontext#"></cfoutput>
</cfcatch>
</cftry>
**cfchart**

**Description**

Generates and displays a chart.

**Category**

Data output tags, Extensibility tags

**Syntax**

```xml
<!--- This syntax uses a JSON file to specify the chart style. --->
<cfchart
    format="html"
    style = "JSON filename">
</cfchart>

OR

<!--- This syntax uses the attributes of the cfchart tag to specify the chart style. --->
<cfchart
    alpha = "value between 0 and 1"
    arrows = "JSON string representation"
    aspect3D = "JSON string representation"
    background = "JSON string representation"
    bevel = "JSON string representation"
    border = "JSON string representation"
    backgroundColor = "hexadecimal value|web color"
    chartHeight = "integer number of pixels"
    chartWidth = "integer number of pixels"
    crosshair = "JSON string representation"
    dataBackgroundColor = "hexadecimal value|web color"
    fill = "JSON string representation"
    font = "font name"
    fontBold = "yes|no"
    fontItalic = "yes|no"
    fontSize = "font size"
    foregroundColor = "hexadecimal value|web color"
    format = "flash|jpg|png|html"
    gridlines = "integer number of lines"
    height = "height in pixels"
    ID = "chart identifier"
    labels = "JSON string representation"
    legend = "JSON string representation"
    labelFormat = "number|currency|percent|date"
    marker = "JSON string representation"
    markerSize = "integer number of pixels"
    name = "string"
    pieSliceStyle = "solid|sliced"
    plot = "JSON string representation"
    plotarea = "JSON string representation"
    preview = "JSON string representation"
    refresh = "canvas|flash|svg|vml"
    renderer = "canvas|flash|svg|vml"
    scales = "comma-separated list of axes"
```
scaleFrom = "integer minimum value"
scaleTo = "integer maximum value"
seriesPlacement = "default|cluster|stacked|percent"
show3D = "yes|no"
showBorder = "yes|no"
showLegend = "yes|no"
showMarkers = "yes|no"
showXGridlines = "yes|no"
showYGridlines = "yes|no"
sortXAxis = "yes|no"
tipBGColor = "hexadecimal value|web color"
tipStyle = "MouseDown|MouseOver|none"
title = "title of chart"
tooltip = "JSON string representation"
url = "onClick destination page"
xAxis = "JSON string representation"
xAxis2 = "JSON string representation"
xAxisTitle = "title text"
xAxisType = "scale|category"
xAxisValues = "Array of values"
xOffset = "number between -1 and 1"
yAxis = "JSON string representation"
yAxis2 = "JSON string representation"
yAxisTitle = "title text"
yAxisType = "scale|category"
yAxisValues = "Array of values"
yOffset = "number between -1 and 1"
Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfchartdata, cfchartseries, cfdocument. Controlling chart appearance in the Developing ColdFusion Applications

History

ColdFusion 11:

The server-side charting introduced in ColdFusion 10 that allowed you to create highly interactive charts has been further enhanced to produce visually more appealing charts.

Note that the new look and feel will not be available for the following tag attributes:

cfchart

- labelFormat
- seriesPlacement with percent as the value
- sortXAxis
- tipStyle
- xAxisType
- yAxisType
- xOffset
- yOffset

cfchartseries

- markerStyle: letterx, mcross, snow, and rcross.
- paintStyle

ColdFusion provides a utility to convert XML styles (used by the old charting system) to JSON styles (used by the new charting system).

To perform this conversion, you need to use cfchart_xmltojson.bat (or cfchart_xmltojson.sh) available in <cfusion_home>/bin folder:

```
cfchart_xmltojson.bat <xml_file_path>
```

The converted JSON style will be created in the same location as the XML file.
Also, in ColdFusion 11, the XML format style has been removed.

**ColdFusion 10:** Added the following new attributes: `alpha`, `arrows`, `aspect3D`, `background`, `bevel`, `border`, `crosshair`, `fill`, `format`, `height`, `ID`, `labels`, `legend`, `plot`, `plotarea`, `preview`, `refresh`, `renderer`, `scales`, `type`, `tooltip`, `width`, `xaxis`, `axis2`, `xaxisvalues`, `yaxis`, `yaxis2`, `yaxisvalues`, `zoom`

**ColdFusion 8:**
- Added the new attribute `showLegend` to the chart style files, which are the XML files located in the `charting/styles` directory. This attribute displays an entry for each point and is applicable only to charts that contain a single series. By default, the value of `showLegend` is set to `true`. To turn off this feature, you can either modify the setting in all the chart style files, or use a custom style file.

**ColdFusion MX 7.01:** Changed documentation to state that the `fontSize` attribute can accept a number that is not an integer.

**ColdFusion MX 7:**
- Added `style` and `title` attributes.
- Added support for eight-digit hexadecimal values to specify RGB color and transparency.
- Removed the `rotated` attribute.

**ColdFusion MX 6.1:**
- Added the `xAxisType` and `yAxisType` attributes.
- Changed interpolation behavior: the tag now interpolates data points on line charts with multiple series.

ColdFusion MX: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>Optional</td>
<td>Alpha (transparency) level of the background. Valid values range from 0 (transparent) to 1 (opaque).</td>
<td></td>
</tr>
<tr>
<td>arrows</td>
<td>Opt</td>
<td></td>
<td>Creates an arrow for pointing out data or other chart items. JSON string representation of array of structs that contain values such as <code>to</code>, <code>from</code>, <code>size</code>, and <code>label</code>.</td>
</tr>
<tr>
<td>aspect3D</td>
<td>Opt</td>
<td>JSON string representation of struct that defines the angle of 3D aspect. The valid struct keys are <code>angle</code> and <code>depth</code>.</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>background</td>
<td>Opt</td>
<td>A struct of keys related to background such as</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>color</code>: Sets the background color(s)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>color-1</code>: Sets the first background color for the arrow</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>color-2</code>: Sets the second background color for the arrow (used with gradients)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>transparent</code>: Set the transparency of a background image so that underlying colors or chart can show.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>fit</code>: Defines the width/height to fit area of background.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>repeat</code>: Defines type of image repeat.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>image</code>: Defines the path to the background image.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>position</code>: Defines the position of the background image.</td>
<td></td>
</tr>
<tr>
<td>backgroundColor</td>
<td>Optional</td>
<td>Color of the area between the data background and the chart border, around labels and around the legend. Hexadecimal value or supported named color; see the name list in Usage. For a hexadecimal value, use the form &quot;##x xxxxx&quot; or &quot;##xxxxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none.</td>
<td></td>
</tr>
</tbody>
</table>
| bevel       | Opt | A struct of keys related to bevel such as.  
|            |     | • **color**: Defines the color of the bevel.  
|            |     | • **blur-x**: Defines the sharpness/smoothness of the bevel edges in the x-direction.  
|            |     | • **blur-y**: Defines the sharpness/smoothness of the bevel edges in the y-direction.  
|            |     | • **angle**: Defines the angle of the bevel.  
|            |     | • **distance**: Distance in # | #px indicating the distance from the object the bevel should be displayed.  
| border     | Opt | A struct of keys related to border such as:  
|            |     | • **color**: Sets the color of the border.  
|            |     | • **radius**: Defines the radius of rounded corners.  
|            |     | • **width**: Defines the width of the border.  
| chartHeight| Optional | 240 | Chart height; integer number of pixels.  
| chartWidth | Optional | 320 | Chart width; integer number of pixels.  
| crosshair  | Opt | A struct of keys related to crosshair such as:  
|            |     | • **line-color**: Sets the color of the crosshair lines.  
|            |     | • **alpha**: Defines the alpha transparency level of the line.  
|            |     | • **line-style**: Defines the line style.  

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dataBackgroundColor</code></td>
<td>Optional</td>
<td>white</td>
<td>Color of area around chart data. Hexadecimal value or supported named color; see the name list in the Usage section. For a hexadecimal value, use the form &quot;##xxxxxx&quot; or &quot;##xxxxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none.</td>
</tr>
</tbody>
</table>
| `fill` | Opt | | A struct of keys related to fill such as:  
  - `angle`: Sets the angle at which a linear fill is displayed. A fill angle of zero displays a vertical gradient from top (background-color-1) to bottom (background-color-2).  
  - `offset-x`: Set x-axis offset for background gradient.  
  - `offset-y`: Set y-axis offset for background gradient. |
| `format` | Req | flash | Format of the chart to be rendered. The supported formats are html, flash, jpg, and png. |
| `font` | Optional | arial | Name of text font:  
  - arial  
  - times  
  - courier  
  - arialunicodeMS.  
  This option is required, if you are using a double-byte character set on UNIX, or using a double-byte character set in Windows with a file type of Flash. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fontBold</td>
<td>Optional</td>
<td>no</td>
<td>Whether to make the text bold:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td>fontItalic</td>
<td>Optional</td>
<td>no</td>
<td>Whether to make the text italicized:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td>fontSize</td>
<td>Optional</td>
<td>11</td>
<td>Font size. If the number is not an integer, ColdFusion rounds the number up to the next integer.</td>
</tr>
<tr>
<td>foregroundColor</td>
<td>Optional</td>
<td>black</td>
<td>Color of text, grid lines, and labels. Hexadecimal value or supported named color; see name list in the Usage section. For a hexadecimal value, use the form &quot;##xxxxxx&quot; or &quot;##xxxxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none.</td>
</tr>
<tr>
<td>format</td>
<td>Optional</td>
<td>flash</td>
<td>File format in which to save the graph:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- jpg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- png</td>
</tr>
<tr>
<td>gridlines</td>
<td>Optional</td>
<td>10, including top and bottom</td>
<td>Number of grid lines to display on the value axis, including axis; positive integer.</td>
</tr>
<tr>
<td>height</td>
<td>Opt</td>
<td></td>
<td>Chart height; integer; number of pixels.</td>
</tr>
<tr>
<td>ID</td>
<td>Opt</td>
<td></td>
<td>ID of the chart. Used to get the underlying chart object.</td>
</tr>
<tr>
<td>labels</td>
<td>Opt</td>
<td>An array of structs used to display custom text or images on the chart, for example author or chart information.</td>
<td></td>
</tr>
</tbody>
</table>
| labelFormat | Optional | number | Format for y-axis labels:  
- number  
- currency  
- percent  
- date |
| legend | Opt | A struct used to define the legend attributes, for example, background-color or margin-top. |
| markerSize | Optional | (Automatic) | Size of data point marker in pixels; integer. |
| name | Optional | | Page variable name; string. Generates the graph as binary data and assigns it to the specified variable. Suppresses chart display. You can use the name value in the cffile tag to write the chart to a file. |
| pieSliceStyle | Optional | sliced | Applies to the cfcharts series type attribute value pie.  
- solid: displays pie as if unsliced.  
- sliced: displays pie as if sliced. |
<p>| plot | Opt | A struct of keys such as animation, aspect, margin, and marker used to style the plotting. |
| plotarea | Opt | A struct of keys such as position and margin used to style the plotarea. |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>preview</td>
<td>Opt</td>
<td>A struct of keys such as visible and margin to control chart preview.</td>
</tr>
<tr>
<td>refresh</td>
<td>Opt</td>
<td>A struct of keys such as type, url, and interval to create dynamic charts.</td>
</tr>
<tr>
<td>renderer</td>
<td>Opt</td>
<td>Specify the rendering method. You must remove the format=&quot;html&quot; setting and then set the renderer to “flash”. The supported values are flash, svg, and vml.</td>
</tr>
<tr>
<td>scaleFrom</td>
<td>Optional</td>
<td>Determined by data Y-axis minimum value; integer.</td>
</tr>
<tr>
<td>scales</td>
<td>Opt</td>
<td>Comma-separated list of axis against which to plot the chart, for example, x, y, y2.</td>
</tr>
<tr>
<td>scaleTo</td>
<td>Optional</td>
<td>Determined by data Y-axis maximum value; integer.</td>
</tr>
<tr>
<td>seriesPlacement</td>
<td>Optional</td>
<td>default Relative positions of series in charts that have more than one data series.</td>
</tr>
<tr>
<td>show3D</td>
<td>Optional</td>
<td>yes Whether to display the chart with three-dimensional appearance:</td>
</tr>
</tbody>
</table>

- **default**: ColdFusion determines relative positions, based on graph types
  - cluster
  - stacked
  - percent
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showBorder</td>
<td>Optional</td>
<td>no</td>
<td>Whether to display a border around the chart:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td>showLegend</td>
<td>Optional</td>
<td>yes</td>
<td>Whether to display the legend if the chart contains more than one data series:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td>showMarkers</td>
<td>Optional</td>
<td>yes</td>
<td>Whether to display markers at data points in line, curve, and scatter graphs:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td>showXGridlines</td>
<td>Optional</td>
<td>no</td>
<td>Whether to display x-axis gridlines:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td>showYGridlines</td>
<td>Optional</td>
<td>yes</td>
<td>Whether to display y-axis gridlines:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td>sortXAxis</td>
<td>Optional</td>
<td>no</td>
<td>Whether to display column labels in alphabetic order along the x axis:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ignored if the xAxisType attribute is scale.</td>
</tr>
<tr>
<td>style</td>
<td>Optional</td>
<td></td>
<td>XML file or string to use to specify the style of the chart. In ColdFusion 11, the XML format style has been removed.</td>
</tr>
<tr>
<td>title</td>
<td>Optional</td>
<td></td>
<td>Title of the chart.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tipbcolor</td>
<td>Optional</td>
<td>white</td>
<td>Background color of tips. Applies only to Flash format graph files. Hexadecimal value or supported named color; see the name list in the Usage section. For a hexadecimal value, use the form &quot;##xxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none.</td>
</tr>
<tr>
<td>tipStyle</td>
<td>Optional</td>
<td>mouseOver</td>
<td>Determines the action that opens a pop-up window to display information about the current chart element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- mouseDown: display if the user positions the cursor at the element and clicks the mouse. Applies only to Flash format graph files. (For other formats, this option functions the same as mouseOver.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- mouseOver: displays if the user positions the cursor at the element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- none: suppresses display.</td>
</tr>
<tr>
<td>tooltip</td>
<td>Optional</td>
<td></td>
<td>A struct of keys used to style the tool tip such as background, font, or border.</td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
<td></td>
<td>Type of chart.</td>
</tr>
</tbody>
</table>
| **url**  | Optional | **URL to open if the user clicks item in a data series; the onClick destination page. You can specify variables within the URL string; ColdFusion passes current values of the variables.**
|          |          | - `$VALUE$`: the value of the selected row. If none, the value is an empty string.
|          |          | - `$ITEMLABEL$`: the label of the selected item. If none, the value is an empty string.
|          |          | - `$SERIESLABEL$`: the label of the selected series. If none, the value is an empty string.
|          |          | - "$javascript:...": executes a client-side script.
| **width** | Optional | 320 | **Width of the Chart in pixels.**
| **xAxis** | Optional | **A struct of keys used to style x axis such as form at, guide, item, and label.**
| **xAxis2** | Optional | **A struct of keys used to style second x axis such as format, guide, item, and label, which is on the top of the chart.**
| **xAxisTitle** | Optional | **Title that appears on the x axis; text.**
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xAxisType</td>
<td>Optional</td>
<td>Whether the x axis indicates data or is numeric:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- category: The axis indicates the data category. Data is sorted according to the sortXAxis attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- scale: The axis is numeric. All cfchartdata item attribute values must be numeric. The x axis is automatically sorted numerically.</td>
</tr>
<tr>
<td>xAxisvalues</td>
<td>Opt</td>
<td>An array of values to be displayed on x axis.</td>
</tr>
<tr>
<td>xOffset</td>
<td>Optional</td>
<td>Number of units by which to display the chart as angled, horizontally. Applies if show3D=&quot;yes&quot;. The number can be between 1 and 1, where &quot;-1&quot; specifies 90 degrees left and &quot;1&quot; specifies 90 degrees right.</td>
</tr>
<tr>
<td>yaxis</td>
<td>Opt</td>
<td>A struct of keys used to style y axis such as format, guide, item, and label.</td>
</tr>
<tr>
<td>yaxis2</td>
<td>Opt</td>
<td>A struct of keys used to style second y axis such as format, guide, item, and label, which is on the top of the chart.</td>
</tr>
<tr>
<td>yAxisTitle</td>
<td>Optional</td>
<td>Title of the y axis; text.</td>
</tr>
<tr>
<td>yAxisType</td>
<td>Optional</td>
<td>category</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Currently has no effect, as the y axis is always used for data values.</td>
</tr>
<tr>
<td>yaxisvalues</td>
<td>Opt</td>
<td>An array of values to be displayed on y axis.</td>
</tr>
</tbody>
</table>
### yOffset

**Optional**

0.1

Number of units by which to display the chart as angled, vertically. Applies if show3D="yes". The number can be between 1 and 1, where "-1" specifies 90 degrees left and "1" specifies 90 degrees right.

### zoom

**Opt**

A struct of keys to be applied when you zoom the chart such as alpha, background, or bevel.

---

**Usage**

The `cfchart` tag defines a container in which a graph displays: its height, width, background color, labels, and so on. The `cfchartseries` tag defines the chart style in which data displays: bar, line, pie, and so on. The `cfchartdata` tag defines a data point. Data is passed to the `cfchartseries` tag in the following ways:

- As a query
- As data points, using the `cfchartdata` tag
  
  For the `font` attribute value ArialUnicodeMS, the following rules apply:
  
  - In Windows, to permit Flash charts (`type = "flash"`) to render a double-byte character set, select this value.
  - In UNIX, for all `type` values, to render a double-byte character set, select this value.
  - If this value is selected, the `fontBold` and `fontItalic` attributes have no effect.

  The following table lists W3C HTML 4 named color value or hexadecimal values that the `color` attribute accepts:

<table>
<thead>
<tr>
<th>Color name</th>
<th>RGB value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aqua</td>
<td>##00FFFF</td>
</tr>
<tr>
<td>Black</td>
<td>#000000</td>
</tr>
<tr>
<td>Blue</td>
<td>##0000FF</td>
</tr>
<tr>
<td>Fuchsia</td>
<td>##FF00FF</td>
</tr>
<tr>
<td>Gray</td>
<td>##808080</td>
</tr>
<tr>
<td>Green</td>
<td>#008000</td>
</tr>
<tr>
<td>Lime</td>
<td>##00FF00</td>
</tr>
<tr>
<td>Maroon</td>
<td>##800000</td>
</tr>
<tr>
<td>Color</td>
<td>Hexadecimal Value</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Navy</td>
<td>#000080</td>
</tr>
<tr>
<td>Olive</td>
<td>#808000</td>
</tr>
<tr>
<td>Purple</td>
<td>#800080</td>
</tr>
<tr>
<td>Red</td>
<td>#FF0000</td>
</tr>
<tr>
<td>Silver</td>
<td>#C0C0C0</td>
</tr>
<tr>
<td>Teal</td>
<td>#008080</td>
</tr>
<tr>
<td>White</td>
<td>#FFFFFF</td>
</tr>
<tr>
<td>Yellow</td>
<td>#FFFF00</td>
</tr>
</tbody>
</table>

For all other color values, enter the hexadecimal value. You can enter a six-digit value, which specifies the RGB value, or an eight-digit value, which specifies the RGB value and the transparency. The first two digits of an eight-digit hexadecimal value specify the degree of transparency, with FF indicating opaque and 00 indicating transparent. Values between 00 and FF are allowed.

For more color names that are supported by popular browsers, go to [www.w3.org/TR/css3-color](http://www.w3.org/TR/css3-color). You can specify whether charts are cached in memory, the number of charts to cache, and the number of chart requests that ColdFusion can process concurrently. To set these options in the ColdFusion Administrator, select Server Settings>Charting.

For client-side charting, the following attributes are not supported: format, labelformat, seriesplacement with percent as the value, sort, xaxis, tipsstyle, url, xAxisType, xoffset, yaxistype, and yoffset.

**Example**
The following example analyzes the salary data in the cfdocexamples database and generates a bar chart showing average salary by department. The body of the cfchartseries tag includes one cfchartdata tag to include data that is not available from the query. --->

Get the raw data from the database. --->
<cfquery name="GetSalaries" datasource="cfdocexamples">
SELECT Departmt.Dept_Name,
Employee.Dept_ID,
Employee.Salary
FROM Departmt, Employee
WHERE Departmt.Dept_ID = Employee.Dept_ID
</cfquery>

Use a query of queries to generate a new query with statistical data for each department. --->
AVG and SUM calculate statistics. --->
GROUP BY generates results for each department. --->
<cfquery dbtype = "query" name = "DataTable">
SELECT Dept_Name,
AVG(Salary) AS avgSal,
SUM(Salary) AS sumSal
FROM GetSalaries
GROUP BY Dept_Name
</cfquery>

Reformat the generated numbers to show only thousands. --->
<cfloop index = "i" from = "1" to = "#DataTable.RecordCount#">
<cfset DataTable.sumSal[i] = Round(DataTable.sumSal[i]/1000)*1000>
<cfset DataTable.avgSal[i] = Round(DataTable.avgSal[i]/1000)*1000>
</cfloop>

Employee Salary Analysis</h1>

Bar graph, from Query of Queries --->
<cfchart format="flash"
xaxistitle="Department"
yaxistitle="Salary Average">
<cfchartseries type="bar"
query="DataTable"
itemcolumn="Dept_Name"
valuecolumn="avgSal">
<cfchartdata item="Facilities" value="35000">
</cfchartseries>
</cfchart>
**cfchartdata**

**Description**

Used with the `cfchart` and `cfchartseries` tags. This tag defines chart data points. Its data is submitted to the `cfchartseries` tag.

**Category**

Data output tags, Extensibility tags

**Syntax**

```xml
<cfchartdata
    item = "text"
    value = "number">
</cfchartdata>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

`cfchart`, `cfchartseries`; Creating Charts and Graphs in the _Developing ColdFusion Applications_ColdFusion MX: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Required</td>
<td></td>
<td>Data point name; string.</td>
</tr>
<tr>
<td>value</td>
<td>Required</td>
<td></td>
<td>Data point value; number or expression.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<cfchart>
    <cfchartseries value="10">
        <cfchartdata item="item1" value="data1"/>
        <cfchartdata item="item2" value="data2"/>
    </cfchartseries>
</cfchart>
```
The following example analyzes the salary data in the cfdocexamples database and generates a bar chart showing average salary by department. The body of the cfchartseries tag loops over a cfchartdata tag to include data available from the query. --->

Get the raw data from the database. --->
<cfquery name="GetSalaries" datasource="cfdocexamples">
SELECT Departmt.Dept_Name,
    Employee.Dept_ID,
    Employee.Salary
FROM Departmt, Employee
WHERE Departmt.Dept_ID = Employee.Dept_ID
</cfquery>

Use a query of queries to generate a new query with --->
statistical data for each department. --->
AVG and SUM calculate statistics. --->
GROUP BY generates results for each department. --->
<cfquery dbtype = "query" name = "DataTable">
SELECT Dept_Name,
    AVG(Salary) AS avgSal,
    SUM(Salary) AS sumSal
FROM GetSalaries
GROUP BY Dept_Name
</cfquery>

Reformat the generated numbers to show only thousands. --->
<cfloop index = "i" from = "1" to = "#DataTable.RecordCount#">
<cfset DataTable.sumSal[i] = Round(DataTable.sumSal[i]/1000)*1000>
<cfset DataTable.avgSal[i] = Round(DataTable.avgSal[i]/1000)*1000>
</cfloop>

Employee Salary Analysis</h1>
Bar graph, from Query of Queries. --->
<cfchart format="flash"
xaxistitle="Department"
yaxistitle="Salary Average">
<cfchartseries type="bar">
itemcolumn="Dept_Name"
valuecolumn="avgSal"
</cfchartseries>
</cfchart>
**cfchartseries**

**Description**

Used with the `cfchart` tag. This tag defines the chart style in which the data displays: bar, line, pie, and so on.

**Category**

*Data output tags*, *Extensibility tags*

**Syntax**

```xml
<cfchartseries
alpha = "Integer between 0 and 1"
animate = "JSON string representation"
aspect = "text"
background = "JSON string representation"
bevel = "JSON string representation"
border = "JSON string representation"
color = "text"
data = "JSON string representation"
hovermarker = "JSON string representation"
marker = "JSON string representation"
type = "type"
itemColumn = "query column"
label = "text"
valueColumn = "query column"
colorlist = "list"
dataLabelStyle = "style"
markerStyle = "style"
paintStyle = "plain|raise|shade|light"
query = "query name"
scales = "comma-separated list of axes"
shadow = "JSON string representation"
seriesColor = "hexadecimal value|web color"
seriesLabel = "label text">
tooltip = "JSON string representation"
zColumn = "query column"
type = "type"
itemColumn = "query column"
valueColumn = "query column"
colorlist = "list"
dataLabelStyle = "style"
markerStyle = "style"
paintStyle = "plain|raise|shade|light"
query = "query name"
seriesColor = "hexadecimal value|web color"
seriesLabel = "label text">
</cfchartseries>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.
See also

cfchart, cfchartdata; Creating Charts and Graphs in the Developing ColdFusion Applications

History

ColdFusion 10: Added the following attributes: {{alpha, animate, aspect, background, bevel, border, marker, color, label, hoverMarker, data, scales, shadow, tooltip, zcolumn}}ColdFusion MX 7:

- Added the dataLabelStyle attribute.
- Added the horizontalbar value of the type attribute.

ColdFusion MX 6.1: Changed interpolation behavior: the tag now interpolates data points on line charts with multiple series.ColdFusion MX: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>Optional</td>
<td></td>
<td>Alpha (transparency) level of the background. Valid values range from 0 (transparent) to 1 (opaque) .</td>
</tr>
<tr>
<td>animate</td>
<td>Opt</td>
<td></td>
<td>A struct of keys to define the animation such as effect and speed. An empty struct results in default animation with appear effect.</td>
</tr>
<tr>
<td>aspect</td>
<td>Opt</td>
<td></td>
<td>Defines the variations of a chart type, for example line, area, and dots in a radar chart.</td>
</tr>
</tbody>
</table>
| background | Opt | A struct of keys related to background such as:
|           |    | - **color**: Sets the background color(s)
|           |    | - **color-1**: Sets the first background color for the arrow
|           |    | - **color-2**: Sets the second background color for the arrow (used with gradients)
|           |    | - **transparent**: Set the transparency of a background image so that underlying colors or chart can show.
|           |    | - **fit**: Defines the width/height to fit area of background.
|           |    | - **repeat**: Defines type of image repeat.
|           |    | - **image**: Defines the path to the background image.
|           |    | - **position**: Defines the position of the background image.

| bevel     | Opt | A struct of keys related to bevel such as:
|           |    | - **color**: Defines the color of the bevel.
|           |    | - **blur-x**: Defines the sharpness/smoothness of the bevel edges in the x-direction.
|           |    | - **blur-y**: Defines the sharpness/smoothness of the bevel edges in the y-direction.
|           |    | - **angle**: Defines the angle of the bevel.
<p>|           |    | - <strong>distance</strong>: Distance in # | #px indicating the distance from the object the bevel should be displayed. |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| **border** | Opt | A struct of keys related to border such as:  
  - color: Sets the color of the border.  
  - radius: Defines the radius of rounded corners.  
  - width: Defines the width of the border. |
| **marker** | Opt | A struct of keys used to style the marker such as size, border, background, and bevel. |
| **color** | Opt | Color of the main element (such as the bars) of a chart.  
For a pie chart, this is the color of the first slice.  
Hexadecimal value or supported named color; see the name list and information about six- and eight-digit hexadecimal values in the Usage section for the cfchart tag.  
For a hexadecimal value, use the form "##xxxxxx" or "##xxxxxxxx", where x = 0-9 or A-F; use two number signs or none. |
| **label** | Opt | Text of the data series label. |
| **hoverMarker** | Opt | A struct of keys used to style the marker on mouse hover such as size, border, background, and bevel. |
| **data** | Opt | The chart data. This is an array of arrays. Specify the data as follows: `["a", 3], ["b", 5]` or `["a", 4, 5], ["b", 7, 3]` for charts with z parameter such as bubble chart. |
| **scales** | Opt | Comma-separated list of axis against which to plot the series, for example, x,y2. |
| **shadow** | Opt | false | Used to enable/disable the shadow. The value can either be yes|no or a struct that takes the key alpha. |
| **tooltip** | Opt | A struct of keys used to style the tool tip such as background, font, or border. |
| **zcolumn** | Opt | The value of the dimension. Applicable if the chart takes a third dimension in addition to x and y. |
| **type** | Required | Sets the chart display style:  
- bar  
- line  
- pyramid  
- area  
- horizontalbar  
- cone  
- curve  
- cylinder  
- step  
- scatter  
- pie |
<p>| <strong>itemColumn</strong> | Required if query attribute is specified | Name of a column in the query specified in the query attribute; contains the item label for a data point to graph. |</p>
<table>
<thead>
<tr>
<th><strong>valueColumn</strong></th>
<th>Required if query attribute is specified</th>
<th>Name of a column in the query specified in the query attribute; contains data values to graph.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>colorlist</strong></td>
<td>Optional</td>
<td>Sets colors for each data point. Applies if the <code>cfchart</code> <code>chartseries type</code> attribute is <code>pie, pyramid, area, h orizontalbar, cone, cylinder, or step.</code> Comma-delimited list of hexadecimal values or supported, named web colors; see the name list and information about six- and eight-digit hexadecimal values in the <code>cfchart</code> Usage section. For a hexadecimal value, use the form &quot;##xxxxxx&quot; or &quot;##xxxxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none.</td>
</tr>
</tbody>
</table>
| **dataLabelStyle** | Optional | none | Specifies the way in which the color is applied to the item in the series:  
- **none**: nothing is printed.  
- **value**: the value of the datapoint.  
- **rowLabel**: the row's label.  
- **columnLabel**: the column's label.  
- **pattern**: combination of column label, value, and aggregate information, such as the `columnLabel value` for the percentage of total graph, for example, `Sales 55,000 20% of 277,000`. |
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| markerStyle     | Optional | rectangle    | Sets the icon that marks a data point for two-dimensional line, curve, and scatter graphs:  
|                 |       |               | • rectangle  
|                 |       |               | • triangle  
|                 |       |               | • diamond  
|                 |       |               | • circle  
|                 |       |               | • letter  
|                 |       |               | • mcross  
|                 |       |               | • snow  
|                 |       |               | • rcross |
| paintStyle      | Optional | plain         | Sets the paint display style of the data series:  
|                 |       |               | • plain: solid color.  
|                 |       |               | • raise: the appearance of a button.  
|                 |       |               | • shade: gradient fill, darker at the edges.  
|                 |       |               | • light: a lighter shade of color; gradient fill. |
| query           | Optional |               | Name of the ColdFusion query from which to get data to graph. |
| seriesColor     | Optional |               | Color of the main element (such as the bars) of a chart. For a pie chart, the color of the first slice. Hexadecimal value or supported named color; see the name list and information about six- and eight-digit hexadecimal values in the Usage section for the `<cfchart>` tag. For a hexadecimal value, use the form "##xxxxxx" or "##xxxxxxxx", where x = 0-9 or A-F; use two number signs or none. |
| seriesLabel     | Optional |               | Text of the data series label |

**Usage**
The following attributes are not supported for Client side charting: `paintStyle` and the following values for `marker style`: `letterx`, `mcross`, `snow`, and `rcross`. A new attribute `zvalue` added to `cfchartdata`: Applicable if chart takes a third dimension in addition to x and y.

For a pie chart, ColdFusion sets pie slice colors as follows:

- If the `seriesColor` attribute is omitted, ColdFusion automatically determines the colors of the slices.
- If the `seriesColor` attribute is specified, ColdFusion automatically determines the colors of the slices after the first one, starting with the specified color for the first slice.

Limitations

The following server-side charting features are not available with client-side charting:

- Linking charts to URL
- Writing charts to a variable

Example 1
Example 2

The following is a basic example of using client side charting.
Example 3

This example showcases how you can create a simple bubble chart by specifying zcolumn.

```xml
<cfchart format="html" type="bubble" query="ChartQuery" showlegend="false">
  <cfchartseries query="ProdQuery" type="bubble" itemcolumn="title" valuecolumn="total_days" zcolumn="rem_days" label="Total_Days">
    <cfchartdata item=1 value=10 zvalue=40>
    <cfchartdata item=2 value=20 zvalue=30>
    <cfchartdata item=3 value=30 zvalue=20>
    <cfchartdata item=4 value=20 zvalue=35>
    <cfchartdata item=5 value=40 zvalue=10>
  </cfchartseries>
</cfchart>
```

Example 4

This is an example that specifies labels as a struct.

```xml
<cfchart format="html" type="bubble" query="ChartQuery" showlegend="false" labels=#["Hello Label","hook":"node:plot=0,index=2,offset-x=-10,offset-y=-90"]#>
  <cfchartseries type="bubble" label="Total_Days">
    <cfchartdata item=1 value=10 zvalue=40>
    <cfchartdata item=2 value=20 zvalue=30>
    <cfchartdata item=3 value=30 zvalue=20>
    <cfchartdata item=4 value=20 zvalue=35>
    <cfchartdata item=5 value=40 zvalue=10>
  </cfchartseries>
</cfchart>
```
**cfcol**

**Description**

Defines table column header, width, alignment, and text. Used within a `cftable` tag.

**Category**

Data output tags

**Syntax**

```xml
<cfcol
    header = "column header text"
    text = "column text"
    align = "left|right|center"
    width = "number that indicates width of column">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfcontent, cfoutput, cftable; Performing file operations with cfftp in the Developing ColdFusion Applications

**History**

ColdFusion MX: Added the ability to construct dynamic cfcol statements.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
<td>Required</td>
<td></td>
<td>Column header text. To use this attribute, you must also use the <code>cftable.colHeaders</code> attribute.</td>
</tr>
<tr>
<td>text</td>
<td>Required</td>
<td></td>
<td>Double-quoted mark-delimited text; determines what to display. Rules: same as for cfoutput sections. You can embed hyperlinks, image references, and input controls.</td>
</tr>
<tr>
<td>align</td>
<td>Optional</td>
<td>left</td>
<td>Column alignment:</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• left</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• center</td>
</tr>
<tr>
<td>width</td>
<td>Optional</td>
<td>20</td>
<td>Column width. If the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>length of data displayed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>exceeds this value, data is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>truncated to fit. To avoid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>this, use an HTML table</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tag. If the surrounding cftabl e tag includes the htmlt able attribute, width specifies the percent of the table width and it does not truncate text; otherwise, w idth specifies the number of characters.</td>
</tr>
</tbody>
</table>

**Usage**

At least one cfcol tag is required within the cftable tag. You must put cfcol and cftable tags adjacent in a page. The only tag that you can nest within the cftable tag is the cfcol tag. You cannot nest cftable tags. To display the cfcol header text, specify the cfcol header and the cftable colHeader attribute. If you specify either attribute without the other, the header does not display. No error is thrown.

**Example**
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
  SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
  FROM Employees
</cfquery>
<html>
<body>
<h3>cfcol Example</h3>
<!--- Uses the HTMLTable attribute to display cftable as an HTML table, rather than PRE formatted information --->
<cftable
  query = "GetEmployees"
  startRow = "1" colSpacing = "3"
  HTMLTable colheaders>
  <!--- Each cfcol tag sets the width of a column in the table, the header information, and the text/CFML for the cell. --->
  <cfcol header = "<b>ID</b>"
    align = "Left"
    width = 2
    text= "#Emp_ID#">
  <cfcol header = "<b>Name/Email</b>"
    align = "Left"
    width = 15
    text= "<a href = mailto:#Email#>#FirstName# #LastName#</A>">
  <cfcol header = "<b>Phone Number</b>"
    align = "Center"
    width = 15
    text= "#Phone#">
</cftable>
cfcollection

Description

Creates and administers Solr search engine collections.

Category

Extensibility tags

Syntax

```
collection supports script style syntax:

new collection().CREATE(collection="<collection_name>", engine="solr", path="<path to the solr directory>");
```

```
<cfcollection
  action = "action"
  categories = "yes|no"
  collection = "collection name"
  engine = "solr"
  language = "language"
  name = "query name"
  path = "c">
```

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfexecute, cfindex, cfobject, cfreport, cfsearch, cfwddx

History

ColdFusion 9: Added the engine attribute (required for Solr support).
ColdFusion MX 7:

- Starting with ColdFusion MX 7, you cannot use the cfcollection tag to create alias names for existing collections. Because Verity maintains all the collection information, you cannot have two names point to the same collection.
- Removed reference to external collections.
- Deprecated the map and repair options of the action attribute. They might not work, and might cause an error, in later releases.
- Added categories attribute and categorylist action.
- Added CATEGORIES, SIZE, DOCCOUNT, and LASTMODIFIED to list of variables returned by the list action.
- Marked as obsolete the MAPPED, ONLINE, and REGISTERED variables returned by the list action.
ColdFusion MX:

- Changed the requirements for the action attribute: it is now required.
- Added the `action` attribute `list` value. It is the default.
- Changed the requirements for the `action` attribute value `map`: it is not necessary to specify the `action` attribute value `map`. (ColdFusion detects collections and creates maps collections as required.)
- Changed acceptable collection naming: this tag accepts collection names that include spaces.
- Changed Verity operations behavior: ColdFusion supports Verity operations on Acrobat PDF files.
- Changed thrown exceptions: this tag can throw the SEARCHENGINE exception.

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| `action`  | Required; see Usage | `list`  | • `categorylist`: Retrieves categories from the collection and indicates how many documents are in each one. Returns a structure of structures in which the category representing each substructure is associated with a number of documents. For a category in a category tree, the number of documents is the number at or below that level in the tree.  
• `create`: registers the collection with ColdFusion. If the collection is present, the tag creates a map to it. If the collection is not present, the tag creates it.  
• `delete`: unregisters a collection and deletes its directories.  
• `list`: returns a query result set, named from the `name` attribute value, of the attributes of the collections that are registered by ColdFusion. If you have Solr collections and omit the `engine` attribute, ColdFusion lists information for both types of collections. |
<table>
<thead>
<tr>
<th><strong>Action</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>map</strong></td>
<td>creates a map to a collection. If the action is <code>create</code> and the collection exists, ColdFusion also creates a map to the collection.</td>
</tr>
<tr>
<td><strong>optimize</strong></td>
<td>optimizes the structure and contents of the collection for searching; recovers space. Causes collection to be taken offline, preventing searches and indexing.</td>
</tr>
<tr>
<td><strong>repair</strong></td>
<td>deprecated. Does nothing.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>categories</strong></th>
<th>See Usage</th>
<th>no</th>
<th>Used only for creating a collection:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>yes</strong>: this collection includes support for categories.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>no</strong>: this collection does not support categories.</td>
</tr>
</tbody>
</table>

| **collection** | See Usage |    | • A collection name. The name can include spaces. |

<table>
<thead>
<tr>
<th><strong>engine</strong></th>
<th>Optional</th>
<th>Solr</th>
<th>The search engine for the collection:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>solr</strong>: the Apache Lucene open source search engine</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For the <code>create</code> and <code>map</code> actions, the default is Solr.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For the <code>list</code> action, the default is to list all collections, Solr. For all other actions ColdFusion determines the collection type.</td>
</tr>
</tbody>
</table>

<p>| <strong>language</strong> | See Usage | English | For a list of options, see Usage. |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>See Usage</th>
<th>Name for the query results returned by the list and categorylist actions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>See Usage</td>
<td>Absolute path to a collection. To map an existing collection, specify a fully qualified path to the collection (not including the collection name); for example, &quot;C:\MyCollections&quot;.</td>
</tr>
</tbody>
</table>

**Usage**

With this tag you can

- Create Solr collections.
- Administer Solr collections created by this tag or the ColdFusion Administrator.

The following table shows the dependence relationships among this tag's attribute values:

<table>
<thead>
<tr>
<th>This attribute is required, optional, or unnecessary (blank):</th>
<th>For this action attribute value:</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>create</td>
</tr>
<tr>
<td>collection</td>
<td>Required</td>
</tr>
<tr>
<td>path</td>
<td>Required</td>
</tr>
<tr>
<td>language</td>
<td>Optional</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
</tr>
<tr>
<td>categories</td>
<td></td>
</tr>
</tbody>
</table>

The following examples show the structures returned by the categorylist action:

**CATEGORIES**
The **list** action returns the following information in a result set that contains one row per collection:

<table>
<thead>
<tr>
<th>Column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORIES</td>
<td>- <strong>yes</strong>: the collection has category support enabled.</td>
</tr>
<tr>
<td></td>
<td>- <strong>no</strong>: the collection does not have category support enabled.</td>
</tr>
<tr>
<td>CHARSET</td>
<td>The character set of the collection.</td>
</tr>
<tr>
<td>CREATED</td>
<td>The date and time that the collection was created.</td>
</tr>
<tr>
<td>DOCCOUNT</td>
<td>The number of documents in this collection.</td>
</tr>
<tr>
<td>EXTERNAL</td>
<td>- <strong>yes</strong>: the collection is external.</td>
</tr>
<tr>
<td></td>
<td>- <strong>no</strong>: the collection is not external.</td>
</tr>
<tr>
<td></td>
<td>- <strong>not found</strong>: the collection is registered but is not available in the defined path.</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>The locale setting of the collection. This information is not available for K2Server collections.</td>
</tr>
<tr>
<td>LASTMODIFIED</td>
<td>The date and time that the collection was last changed.</td>
</tr>
<tr>
<td>MAPPED</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>NAME</td>
<td>The name of the collection.</td>
</tr>
<tr>
<td>ONLINE</td>
<td>Obsolete.</td>
</tr>
</tbody>
</table>
### PATH
Absolute path to the collection.

### REGISTERED
Obsolete.

### SIZE
The size of the collection, expressed in kilobytes.

You can also specify `uni` to enable support for multiple languages.

To determine whether a collection exists, use code, such as the following, to execute a query of queries:

```cfcollection action="list" name="myCollections" >
<cfquery name="qoq" dbtype="query">
SELECT * from myCollections
WHERE myCollections.name = 'myCollectionName'
</cfquery>
<cfif qoq.recordcount GT 0>
<!--- Collection exists --->
<cfdump var = #qoq#>
</cfif>
```

To determine whether a Solr collection exists, you must specifically add the attribute `engine` and provide the value as `solr`. For example,`{<cfcollection action="list" name="myCollections" engine="solr">}`To get a result set with values for all the collections that are registered with the search server, use code such as the following:

```cfcollection action="list" name="myCollections">
<cfoutput query="myCollections">
#name#<br>
</cfoutput>
```

To add content to a collection, use `cfindex`. To search a collection, use `cfsearch`. Restart the ColdFusion Search Service for this change to take effect.

For Solr collections, the `language` attribute of this tag supports the following options:

<table>
<thead>
<tr>
<th>Brazilian</th>
<th>cjk (Chinese, Japanese, Korean)</th>
<th>French</th>
<th>Russian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>Dutch</td>
<td>German</td>
<td>Thai</td>
</tr>
<tr>
<td>Chinese</td>
<td>English</td>
<td>Greek</td>
<td></td>
</tr>
</tbody>
</table>
**cfcomponent**

**Description**

Creates and defines a component object: encloses functionality that you build in CFML and enclose in `cffunction` tags. This tag contains one or more `cffunction` tags that define methods. Code within the body of this tag, other than `cffunction` tags, is executed when the component is instantiated. A component file has the extension CFC and is stored in any directory of an application.

A component method is invoked in the following ways:

- In the `cfinvoke` tag in a ColdFusion page
- In a URL that calls a CFC file and passes a method name as a URL parameter
- In the `cfscript` tag
- As a web service
- From Flash code

**Category**

*Extensibility tags*

**Syntax**

```
<cfcomponent
  accessors = "yes|no"
  autoIndex = "yes|no"
  alias = "ActionScript 3 type alias"
  bindingname = "binding element name"
  displayname = "string"
  extends = "component name"
  hint = "string"
  implements = "ColdFusion interface"
  indexable = "yes|no"
  indexLanguage = "language"
  mappedSuperClass = "yes|no"
  namespace = "default service namespace"
  output = "no value|no|yes"
  porttypename = "port type element name"
  Serializable = "yes|no"
  serviceaddress = "service URL"
  serviceportname = "port element name"
  style = "rpc|document|wrapped"
  wsVersion = "1|2"
  wsdlfile = "path">
  variable declarations
  <cffunction ...>
    ...
  </cffunction>
  ...
  <cffunction ...>
    ...
  </cffunction>
</cfcomponent>
```

**See also**
History

ColdFusion 10: rest, restPath, httpMethod, produces, consumes, indexable, indexLanguage, autoIndex, wsVersion

ColdFusion 9.0.1: Added the attribute (mappedSuperClass)

ColdFusion 9: Added the attributes (serializable and accessors)

ColdFusion 8:
- Added the implements and serviceaddress attributes.
- Added support for the onMissingMethod function.
- Added support for publishing document-literal style web services.
- Added the style, namespace, serviceportname, porttypename, wsdllfile, bindingname, and output attributes.
- Extended functionality for the hint and displayname attributes when publishing document-literal style web services.

ColdFusion MX: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessors</td>
<td>Optional</td>
<td>yes (for persistent CFC) no for others</td>
<td>If set to yes, lets you invoke implicit getters and setters. For persistent CFC, accessors is always enabled.</td>
</tr>
<tr>
<td>alias</td>
<td>Optional</td>
<td></td>
<td>Specifies the type label to give the object when it is converted from CFML to ActionScript 3. It matches the alias attribute of AS3 types. This attribute applies only to Flash Remoting and LiveCycle Data Services value objects, and lets you work with typed objects in both ColdFusion and Flash.</td>
</tr>
<tr>
<td>autoIndex</td>
<td>Optional</td>
<td>yes</td>
<td>If no, auto-indexing of CFC does not occur. That is, indexing occurs only in offline mode.</td>
</tr>
<tr>
<td>bindingname</td>
<td>Optional</td>
<td>Specifies the <code>binding</code> attribute of the <code>port</code> element in the WSDL. If you don't specify this attribute, ColdFusion derives the value from the CFC class name.</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>consumes</td>
<td>optional</td>
<td>Comma-separated list of acceptable MIME types, for example <code>consumes=&quot;text/plain,text/html&quot;</code>. Searches for the Content-Type header in the HTTP request; if the value is, for example HTML or XML, ColdFusion finds the appropriate method which can handle the request and invoke the method. Used to specify which MIME media types of representation a resource can accept or consume, from the client. If this attribute is used at the component level (and not at the function level), all the response methods accept the specified MIME types by default. If no function in a CFC can consume the MIME type in a client request, the following error occurs: 415 Unsupported Media Type. If no value is specified, <code>/</code> is taken by default, which means, all MIME types are consumed.</td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>Optional</td>
<td>A string that displays when you use introspection to show information about the CFC. The information appears on the heading, following the component name.</td>
<td></td>
</tr>
<tr>
<td>extend</td>
<td>Optional</td>
<td>WEB-INF.cftags.component</td>
<td>Name of parent component from which to inherit methods and properties. You can use the keyword component to specify the default value.</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>hint</td>
<td>Optional</td>
<td>Text that displays when you use introspection to show information about the CFC. The hint attribute value appears below the component name heading. Use this attribute to describe the purpose of the parameter.</td>
<td></td>
</tr>
</tbody>
</table>
| httpMethod | optional | The HTTP method to use, must be one of the following:  
  - GET: Requests information from the server. Any data that the server requires to identify the requested information must be in the URL or in the cfhttpparam tag.  
  - POST: Sends information to the server for processing. Requires one or more cfhttpparam tags. Often used for submitting form-like data.  
  - PUT: Requests the server to store the message body at the specified URL. Use this method to send files to the server.  
  - DELETE: Requests the server to delete the specified URL. |
• **HEAD**: Identical to the **GET** method, but the server does not send a message body in the response. Use this method for testing hypertext links for validity and accessibility, determining the type or modification time of a document, or determining the type of server. If you have not specified **HEAD**, by default, **GET** method is called. However, message body is not sent in the response.
• OPTIONS: A request for information about the communication options available for the server or the specified URL. This method enables the ColdFusion application to determine the options and requirements associated with a URL, or the capabilities of a server, without requesting any additional activity by the server. If you have not specified OPTIONS, then ColdFusion sends a response. Based on the request method, the corresponding function is invoked. For example, if a GET request is sent to the server, then a function with httpMethod as GET is invoked. If this value is not specified at the function level, then the value you define here is used. So all functions for which httpMethod is not specified are assigned this value.
<p>| <strong>implements</strong> | Optional | Name of the ColdFusion interface or interfaces that this component implements. If the component implements an interface, it must define all the functions in the interface, and the function definitions must conform to the definitions specified in the interface. For more information, see <code>cfinterface</code>. A component can implement any number of interfaces. To specify multiple interfaces, use a comma-delimited list with the format <code>interface1, interface2</code>. |
| <strong>indexable</strong> | Optional | no | If yes, enables indexing for the component. |
| <strong>indexLanguage</strong> | Optional | english | Specify the language that is used to index and search. The value you set overrides the value defined in the Application.cfc. |
| <strong>mappedSuperClass</strong> | Optional | no | If set to yes on a non-persistent CFC, child CFCs can inherit its properties. For example, you can define a base CFC with common properties such as ID, version, or createdOn which all other persistent CFCs would extend and thus get one common behavior. <code>mappedSuperClass</code> cannot be set to yes on a persistent CFC. |</p>
<table>
<thead>
<tr>
<th><strong>namespace</strong></th>
<th>Optional</th>
<th><strong>class name</strong></th>
<th>Specifies the namespace used in the WSDL for a CFC that is invoked as a web service. If you don’t specify this attribute, ColdFusion derives the value from the CFC class name.</th>
</tr>
</thead>
</table>
| **output** | Optional | Component body displayable text that is processed as standard CFML | Specifies whether constructor code in the component can generate HTML output; does not affect output in the body of cffunction tags in the component.  
- **yes**: Constructor code is processed as if it were within a cfoutput tag. Variable names surrounded by number signs (#) are automatically replaced with their values.  
- **no**: Constructor code is processed as if it were within a cfsilent tag.  
- If you do not specify this attribute, constructor code is processed as standard CFML. Any variables must be in cfoutput tags. |
<p>| <strong>porttypename</strong> | Optional | Specifies the name attribute of the porttype element in the WSDL. If you don’t specify this attribute, ColdFusion derives the value from the CFC class name. |</p>
<table>
<thead>
<tr>
<th>produces</th>
<th>optional</th>
<th>/</th>
<th>Comma-separated list of acceptable MIME types, for example <em>produces</em>=&quot;text/plain,text/html&quot;. Searches for the Accept header in the HTTP request or extension in the URL (valid extensions are .xml and .json); if the value is, for example HTML, JSON, or XML, ColdFusion finds the appropriate method that can handle the request and invoke the method. Used to specify the MIME media types or representations a CFC can produce and send back to the client. If specified at the component level (and not at the function level), all the functions in a CFC can produce the specified MIME types by default. If no methods in a CFC can produce the MIME type in a client request, the following error occurs: 406 Not Acceptable. If no value is specified, / is taken by default, which means, all MIME types are produced.</th>
</tr>
</thead>
<tbody>
<tr>
<td>rest</td>
<td>Optional</td>
<td>true if RestPath is specified</td>
<td>If true, the CFC is REST-enabled.</td>
</tr>
<tr>
<td>restPath</td>
<td>Optional</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
You can access the REST resource either by providing a REST path or CFC path. So specify if you want to use a path other than the CFC path. If rest="true" and you do not specify this attribute, path to the CFC is used. Path to the CFC should be from the directory registered as a REST service. Also, you should include the CFC name in the path. For example, if you have a folder college that is registered as RestTest, and you want to publish student.cfc which is in a sub-folder department (in the folder college), then the URL used to access student.cfc is as follows:

```
http://localhost:8500/rest/RestTest/department/student
```
The restPath in this case is `department/student`. The path is case-sensitive. Also, it is preferable to avoid special character in the path. At CFC level, specify the path as follows:
```
restPath = "restService" or
{"restPath="test/restService"}
```
You can specify templates as value of restPath. For example, `restPath="{name}"`. This means, all URLs in the format `<Base URL>/rest/<service_map/ping>/<any string without slash>` matches the restPath. For example,
```
http://localhost:8500/rest/service1/abc
```
The value in place of template can be accessed using the `restargsource` value. For details, see the attribute details of `cfargument`. The attribute can have complex expressions as values. For example, `restpath="customers/firstname-{lastname}"`. This matches URLs such as `"/customers/paul-bensen" but not "/_customers/paul-bensen". Also `restPath` can include regular expressions such as `restpath="/customers/{id : \d+}"`.
Here, \texttt{id} is an integer that can match "/customers/123111" but not "/customers/asd" or "/customers/123/122".

If you have two methods with different path attribute values and if the paths are ambiguous, then there are precedence rules for finding the match. For example, \texttt{restpath="/customers/{id : .+}". This expression matches any stream of characters after /customer. It matches "/customers/123", "/customers/asd", and "/customers/123/12/asdfa", but not "/customers".

You can also have expressions such as \texttt{restPath="/customers/{id : .+}/address"}, then the URL "/customers/123/asd/address" is matched with both URLs. In such
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serializable</td>
<td>Optional</td>
<td>Specifies whether this component can be serialized. If you set this value to <code>false</code>, the component and the data in the component's <code>This</code> and <code>Variables</code> scopes cannot be serialized, so they are not retained on session replication, and the component is in its default state.</td>
</tr>
<tr>
<td>serviceaddress</td>
<td>Optional</td>
<td>Specifies the SOAP URL of the web service. If you don't specify this attribute, ColdFusion uses the URL of the CFC in the WSDL service description. Use this attribute to specify the protocol, for example, by specifying a URL that starts with <code>https://</code>. This attribute applies only for web services.</td>
</tr>
<tr>
<td>serviceportname</td>
<td>Optional</td>
<td>Specifies the <code>name</code> attribute of the <code>port</code> element in the WSDL. If you don’t specify this attribute, ColdFusion derives the value from the CFC class name.</td>
</tr>
<tr>
<td><strong>tag</strong></td>
<td><strong>Optional</strong></td>
<td><strong>rpc</strong></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>style</strong></td>
<td>Optional</td>
<td>rpc</td>
</tr>
</tbody>
</table>
|         |             | Specifies whether a CFC used for web services uses RPC-encoded style or document-literal style:  
|         |             | - rpc: RPC-encoded style  
|         |             | - document: Document-literal style  
|         |             | - wrapped: If you are setting wsVersion as 2, the default value is wrapped and if it is 1, then the default value is rpc.  

| **wsdlfile** | Optional    | A properly formatted WSDL file to be used instead of WSDL generated by ColdFusion.  

| **wsVersion** | Optional    | If you specify 2, CFC is deployed using Axis 2 engine. The value you specify overrides the value you specify at application or server level.  

### Usage

If you specify the **extends** attribute, the data and methods of the parent component are available to CFC methods as if they were parts of the current component. If the managerCFC component extends the employeeCFC component, and the employeeCFC component has a `getEmployeeName` method, you can call this method by using the managerCFC, as follows:

```cfml
<cfinvoke component="managerCFC" method="getEmployeeName" returnVariable="managerName" EmployeeID=#EmpID#>
```

This tag requires an end tag.

If you specify **style**="document", ColdFusion publishes the CFC as a document-literal style web service. For more information, see [Publishing document-literal style web services](#) in the Developing ColdFusion Applications. CFCs support an `onMissingMethod` function. By defining an `onMissingMethod` function in the `cfcomponent` tag body in the CFC, you can handle calls to methods that are not implemented in the CFC. If an application calls a function that is not defined in the CFC, ColdFusion calls the `onMissingMethod` function and passes it the requested method's name and arguments. If you do not define an `onMissingMethod` function, a call to a method that is not defined in the CFC causes ColdFusion to throw an error that must be handled in the calling code. The `onMissingMethod` function is useful for several purposes:

- To handle errors directly in the component, instead of requiring that each instance of code that calls the
component handles them.

- To create a dynamic proxy, an object that can take arbitrary calls and dynamically determines the correct action.

The `onMissingMethod` function must have the following format:

```html
<cffunction name="onMissingMethod">
    <cfargument name="missingMethodName" type="string">
    <cfargument name="missingMethodArguments" type="struct">
        code to handle call to nonexistent method
    </cffunction>
```

**Note**

The argument name for `onMissingMethod` must not change.

**Example 1**

```html
<cfcomponent>
    <cffunction name="getEmp">
        <cfquery name="empQuery" datasource="cfdocexamples">
            SELECT FIRSTNAME, LASTNAME, EMAIL
            FROM tblEmployees
        </cfquery>
        <cfreturn empQuery>
    </cffunction>

    <cffunction name="getDept">
        <cfquery name="deptQuery" datasource="cfdocexamples">
            SELECT *
            FROM tblDepartments
        </cfquery>
        <cfreturn deptQuery>
    </cffunction>
</cfcomponent>
```
**cfcontent**

**Description**

Does either or both of the following:

- Sets the MIME content encoding header for the current page; if the encoding information includes a character encoding, sets the character encoding of generated output.
- Sends the contents of a file, or of a variable that contains binary data, as the page output.

To restrict this tag, use the settings in the ColdFusion Administrator > Security > Sandbox Security. For more information, see the Administrator online Help.

**Category**

Data output tags

**Syntax**

```<cfcontent
deleteFile = "yes|no"
file = "filename"
reset = "yes|no"
type = "file type"
variable = "variable name"/>
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfcol, cfheader, cfhttp, cfoutput, cftable

**History**

ColdFusion 8: Changed the behavior of the tag if the type attribute is not specified and the file attribute is specified. Previously, ColdFusion assumed a default file type of text/html. Now, ColdFusion attempts to get the content type from the file. ColdFusion MX 7: Added the variable attribute.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| deleteFile | Optional | no    | Applies only if you specify a file with the `file` attribute.  
|            |        |       |  
|            |        |       | - **yes**: deletes the file on the server after sending its contents to the client.  
|            |        |       | - **no**: leaves the file on the server.  
| file       | Optional |       | Name of an on-disk or in-memory file whose contents provide the page output. The filename must start with a drive letter and a colon, or a forward or backward slash. When using ColdFusion in a distributed configuration, the `file` attribute must refer to a path on the system on which the web server runs. When you use this attribute, any other output on the current CFML page is ignored; only the contents of the file are sent to the client.  
| reset      | Optional | yes   | If you specify a `file` or `variable` attribute, this attribute has no effect; otherwise, it does the following:  
|            |        |       |  
|            |        |       | - **yes**: discards output that precedes call to `cf content`  
|            |        |       | - **no**: preserves output that precedes call to `cf content`. In this case, all output is sent with the specified type.  
| type       | Optional |       |             |
The MIME content type of the page, optionally followed by a semicolon and the character encoding. By default, ColdFusion sends pages as text/html content type in the UTF-8 character encoding. However, if the file attribute is specified, ColdFusion attempts to get the content type from the file. The content type determines how the browser or client interprets the page contents. The following are some of the content type values that you can use:
The following list includes commonly used character encoding values:

- utf-8
- iso-8859-1
- windows-1252
- us-ascii
- shift_jis
- iso-2022-jp
- euc-jp
- euc-kr
- big5
- euc-cn
- utf-16

For example:

```json
{{type = "text/html"}}
{{type = "text/html; charset"}}
{{"ISO-8859-1"}}
```
variable | Optional | Name of a ColdFusion binary variable whose contents can be displayed by the browser, such as the contents of a chart generated by the `cfchart` tag or a PDF or Excel file retrieved by a `cffile` tag="readBinary" tag. When you use this attribute, any other output on the current CFML page is ignored; only the contents of the file are sent to the client.

Usage

To set the character encoding (character set) of generated output, including the page HTML, use code such as the following:

```coldfusion
cfcontent type="text/html; charset=ISO-8859-1"
```

When ColdFusion processes an HTTP request, it determines the character encoding to use for the data it returns in the HTTP response. By default, ColdFusion returns character data using the Unicode UTF-8 format, regardless of the value of an HTML `meta` tag in the page. You can use the `cfcontent` tag to override the default character encoding of the response. For example, to tell ColdFusion to return the page using Japanese EUC character encoding, use the `type` attribute, as follows:

```coldfusion
cfcontent type="text/html; charset=EUC-JP"
```

If you call the `cfcontent` tag from a custom tag, and you do not want the tag to discard the current page when it is called from another application or custom tag, set `reset = "no"`. If a file delete operation is unsuccessful, ColdFusion throws an error. Do not use this tag after the `cfflush` tag on a page, it has no effect or ColdFusion throws an error. The following tag can force most browsers to display a dialog box that asks users whether they want to save the contents of the file specified by the `cfcontent` tag using the filename specified by the `filename` value. If the user selects to open the file, most browsers open the file in the related application, not the browser window.

```coldfusion
cfheader name="Content-Disposition" value="attachment; filename=filename.ext"
```

Some file types, such as PDF documents, do not use executable code and can display directly in most browsers. To request the browser to display the file directly, use a `cfheader` tag similar to the following:
<cfheader name="Content-Disposition" value="inline; filename=name.ext">

You can use any value for the *filename* part of the *filename* attribute, but the *ext* part must be the standard Windows extension for the file type. For file types that might contain executable code, such as Microsoft Excel documents, most browsers always ask before opening the document. For these file types, the inline content disposition specification requests the browser to display the file directly if the user selects to open the file.

For more information on character encodings, see the following web pages:

- The page at [www.w3.org/International/O-charset.html](http://www.w3.org/International/O-charset.html) provides general information on character encodings and the web, and has several useful links.
- The page at [www.iana.org/assignments/character-sets](http://www.iana.org/assignments/character-sets) is a complete list of character sets names used on the Internet, maintained by the Internet Assigned Numbers Authority.
- ColdFusion uses the Java JCE for encoding support. The page at [http://java.sun.com/javase/6/docs/technote/s/guides/intl/encoding.doc.html](http://java.sun.com/javase/6/docs/technote/s/guides/intl/encoding.doc.html) lists the character encodings that JCE 6, and therefore ColdFusion, can interpret. This list uses Java internal names, not the IANA character encoding names that you use in the SetEncoding charset parameter and other ColdFusion attributes and parameters.

For a complete list of media types used on the Internet, see [www.iana.org/assignments/media-types/](http://www.iana.org/assignments/media-types/).

⚠️ **Note**

When using `cfabort`, `cflocation`, or `cfcontent` tags, the `OnAbort` method is invoked instead on `OnRequestEnd`.

Example
<!--- CFCONTENT Example 1
This example shows the use of cfcontent to return the contents of the CF Documentation page dynamically to the browser. You might need to change the path and/or drive letter depending on how ColdFusion is installed on your system. Notice that the graphics do not display and the hyperlinks do not work, because the html page uses relative filename references. The root of the reference is the ColdFusion page, not the location of the html page. --->

<cfcontent type = "text/html"
file = "C:\ColdFusion9\wwwroot\cfdocs\dochome.htm"
deleteFile = "no">

<!--- CFCONTENT Example 2
This example shows how the Reset attribute changes text output. Notice how the first text section ("This example shows how the Reset attribute changes output for text reset = "Yes":123) does NOT print out to the screen. --->

<p>This example shows how the Reset attribute changes output for text.</p>
<p>reset = "Yes": 123 <BR> <cfcontent type = "text/html" reset = "Yes">456</p>
<p>This example shows how the Reset attribute changes output for text.</p>
<p>reset = "No": 123 <BR> <cfcontent type = "text/html" reset = "No">456</p>

<!--- CFCONTENT Example 3
This example triggers a download of an Excel file. The user is prompted with an option to save the file or open it in the browser. --->

<cfheader name="Content-Disposition" value="inline; filename=acmesales03.xls">
<cfcontent type="application/vnd.ms-excel" file="c:\temp\acmesales03.xls">

<!--- CFCONTENT Example 4
This example triggers a download of a Word document then deletes the original from the "temp" directory. The user is prompted with an option to save the file or open it in the browser. --->

<cfheader name="Content-Disposition" value="inline; filename=temp.doc">
<cfcontent type="application/msword" file="c:\temp\Cable.doc" deletefile="yes">

<!--- CFCONTENT Example 5
This example causes the browser to treat the HTML table as Excel data. Excel interprets the table format. Because Excel can include executable code, the browser prompts the user whether to save the file or open it in a browser. --->

<cfheader name="Content-Disposition" value="inline; filename=acmesalesQ1.xls">
<cfcontent type="application/vnd.msexcel">
<table border="2">
<tr><td>Month</td><td>Quantity</td><td>$ Sales</td></tr>
<tr><td>January</td><td>80</td><td>$245</td></tr>
<tr><td>February</td><td>100</td><td>$699</td></tr>
<tr><td>March</td><td>230</td><td>$2036</td></tr>
<tr><td>Total</td><td>=Sum(B2..B4)</td><td>=Sum(C2..C4)</td></tr>
</table>
**cfcontinue**

**Description**

Used within a *cfloop* tag. Returns processing to the top of a loop.

**Category**

Flow-control tags

**Syntax**

```
<cfcontinue>
```

**See also**

cfabort, cfbreak, cffexecute, cfif, cflocation, cfloop, cfthrow, cftry; cfloop and cfbreak in the *Developing ColdFusion Applications*

**History**

ColdFusion 9: Added the tag.

**Tag example**

```
<cfloop index="i" from="1" to="5">
<cfoutput>#i#</cfoutput> is
<cfif i mod 2>
  ODD<br>
  <cfcontinue>
</cfif>
  EVEN<br>
</cfloop>
```

**Script example**

```
for (i=1; i <= 5; i++){
  writeOutput("#i# is ");
  if (i mod 2){
    writeOutput("ODD<br>");
    continue;
  }
  writeOutput("EVEN<br>");
}
```

Either version of the code above outputs:
1 is ODD
2 is EVEN
3 is ODD
4 is EVEN
5 is ODD
**cfcookie**

**Description**

Defines web browser cookie variables, including expiration and security options.

**Category**

Forms tags, Variable manipulation tags

**Syntax**

```cfcookie
name = "cookie name"
domain = ".domain"
expires = "period"
httponly = "yes|no"
path = "URL"
secure = "yes|no"
value = "text"
encodevalue = "yes|no"
preserveCase = "yes|no"
>
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfdump, cfparam, cfregistry, cfsavecontent, cfschedule, cfset

**History**

ColdFusion 10: Added the preserveCase and encodeValue attributes.

ColdFusion MX 6.1:

- Changed the expires attribute: it now accepts a date time object.
- Cookie names can include all ASCII characters except commas, semicolons, or whitespace characters.

ColdFusion 9: Added the attribute httponly.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>name</strong></td>
<td><strong>Required</strong></td>
<td>Name of cookie variable. ColdFusion converts cookie names to all-uppercase. Cookie names set using this tag can include any printable ASCII characters except commas, semicolons, or white space characters.</td>
<td></td>
</tr>
<tr>
<td><strong>domain</strong></td>
<td><strong>Required if path attribute is specified. Optional otherwise</strong></td>
<td>Domain in which cookie is valid and to which cookie content can be sent from the user's system. By default, the cookie is only available to the server that set it. Use this attribute to make the cookie available to other servers. Must start with a period. If the value is a subdomain, the valid domain is all domain names that end with this string. This attribute sets the available subdomains on the site on which the cookie can be used. For a domain value that ends in a country code, the specification must contain at least three periods; for example, &quot;.m ongo.state.us&quot;. For top-level domains, two periods are required; for example, &quot;.mgm.com&quot;. You cannot use an IP address as a domain.</td>
<td></td>
</tr>
<tr>
<td><strong>encodevalue</strong></td>
<td><strong>Optional</strong></td>
<td>Specify if cookie value should be encoded</td>
<td></td>
</tr>
<tr>
<td>expires</td>
<td>Optional</td>
<td>session only</td>
<td>Expiration of cookie variable.</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------</td>
<td>--------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The cookie expires when the user closes the browser, that is, the cookie is “session only”.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• A date or date/time object (for example, 10/09/97).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• A number of days (for example, 10, or 100).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• now: deletes cookie from client cookie.txt file (but does not delete the corresponding variable the Cookie scope of the active page).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• never: The cookie expires in 30 years from the time it was created (effectively never in web years).</td>
</tr>
<tr>
<td>httponly</td>
<td>Optional</td>
<td></td>
<td>If yes, sets cookie as httponly so that it cannot be accessed using JavaScripts. Note that the browser must have httponly compatibility.</td>
</tr>
<tr>
<td>path</td>
<td>Optional</td>
<td></td>
<td>URL, within a domain, to which the cookie applies; typically a directory. Only pages in this path can use the cookie. By default, all pages on the server that set the cookie can access the cookie.</td>
</tr>
</tbody>
</table>
|                         |          |              | To specify multiple URLs, use multiple cfcookie tags. If you specify path, also specify domain.
<p>|                         |          |              | path = &quot;/services/login&quot; |</p>
<table>
<thead>
<tr>
<th>preserveCase</th>
<th>Optional</th>
<th>False</th>
<th>Specify if cookie name should be case-sensitive.</th>
</tr>
</thead>
<tbody>
<tr>
<td>secure</td>
<td>Optional</td>
<td></td>
<td>If browser does not support Secure Sockets Layer (SSL) security, the cookie is not sent. To use the cookie, the page must be accessed using the https protocol.</td>
</tr>
<tr>
<td>value</td>
<td>Optional</td>
<td></td>
<td>Value to assign to cookie variable. Must be a string or variable that can be stored as a string.</td>
</tr>
</tbody>
</table>

**Usage**

If this tag specifies that a cookie is saved beyond the current browser session, the client browser writes or updates the cookie in its local cookies file. Until the browser is closed, the cookie resides in browser memory. If the `expires` attribute is not specified, the cookie is not written to the browser cookies file. If you use this tag after the `cfflush` tag on a page, ColdFusion does not send the cookie to the browser; however, the value you set is available to ColdFusion in the Cookie scope during the browser session.

**Note**

You can also create a cookie that expires when the current browser session expires by using the `cfset` tag or a CFScript assignment statement to set a variable in the Cookie scope, as in `<cfset Cookie.mycookie="sugar">`. To get a cookie's value, refer to the cookie name in the Cookie scope, as in `<cfif Cookie.mycookie is "oatmeal">`.

You can use dots in cookie names, as the following examples show:

```
<cfcookie name="person.name" value="wilson, john">
<cfset cookie.person.lastname="Santiago">
```

To access cookies, including cookies that you set and all cookies that are sent by the client, use the Cookie scope. For example, to display the value of the `person.name` cookie set in the preceding code, use the following line:

```
<cfoutput>#cookie.person.name#</cfoutput>
```
<!--- This example shows how to set/delete a cfcookie variable. --->
<!--- Select users who have entered comments into a sample database. --->
<cfquery name = "GetAolUser" dataSource = "cfdocexamples">
  SELECT EMail, FromUser, Subject, Posted
  FROM Comments
</cfquery>
<html>
<body>
<h3>cfcookie Example</h3>
<!--- If the URL variable delcookie exists, set cookie expiration date to NOW --->
<cfif IsDefined("url.delcookie") is True>
  <cfcookie name = "TimeVisited"
    value = "#Now()#"
    expires = "NOW">
</cfif>
<cfelse>
  <!--- Otherwise, loop through list of visitors; stop when you match the string aol.com in a visitor's e-mail address. --->
  <cfloop query = "GetAolUser">
    <cfif FindNoCase("aol.com", Email, 1) is not 0>
      <cfcookie name = "LastAOLVisitor"
        value = "#Email#"
        expires = "NOW">
    </cfif>
  </cfloop>
  <!--- If the timeVisited cookie is not set, set a value. --->
  <cfif IsDefined("Cookie.TimeVisited") is False>
    <cfcookie name = "TimeVisited"
      value = "#Now()#"
      expires = "10">
  </cfif>
</cfelse>
<!--- Show the most recent cookie set. --->
<cfif IsDefined("Cookie.LastAOLVisitor") is "True">
  <p>The last AOL visitor to view this site was #Cookie.LastAOLVisitor#, on #DateFormat(COOKIE.TimeVisited)#</p>
</cfif>
<!--- Use this link to reset the cookies. --->
<p><a href = "cfcookie.cfm?delcookie = yes">Hide my tracks</A></p>
<cfelse>
  <p>No AOL Visitors have viewed the site lately.</p>
</cfif>
The `<cfclient>` tag is a marker tag that instructs ColdFusion to generate client-side code (JavaScript) for ColdFusion code. This tag does not have any attribute. To know more about this tag, read this document.
<cfclientsettings
    enableDeviceAPI = "true|false"
    detectDevice = "true|false"
    deviceTimeout = Number >

See also

cfclient

History

ColdFusion 11: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enableDeviceAPI</td>
<td>Optional</td>
<td>false</td>
<td>Enable/disable the device API.</td>
</tr>
<tr>
<td>detectDevice</td>
<td>Optional</td>
<td>false</td>
<td>Enable/disable the device detection plugin.</td>
</tr>
</tbody>
</table>
### deviceTimeout

**Optional** 10

The timeout for loading the plugins (in seconds).

If the plug-in takes more than 10 seconds (default timeout value) to load, you will get an error. Try increasing the timeout to avoid this problem.

If you have specified `enableDeviceAPI` as "true", the device plugins will be loaded. After the device plugins are loaded, ColdFusion waits for an event "DeviceReady" to be triggered. Once the DeviceReady event is triggered successfully, the execution of CFML starts. In a desktop browser, the DeviceReady event will not be triggered. However, for a mobile application (shell application or packaged application), the event is triggered. So if you are trying to run a CFM where the device APIs are enabled on a desktop browser you will get a blank page and in the console you can see an error message about `deviceTimeout`.

---

**Device detection**

The device detection feature of CFML allows you to identify the device properties and characteristics, which can be used to determine the best content, layout, mark-up or application to serve to the given device.

These characteristics include screen size, browser type and version, media support, and the level of support for CSS, HTML, and JavaScript.

For getting the device features and capabilities, you need to specify an attribute `detectDevice` in the `<cfclientsettings>` tag and set it to true:

```
<cfclientsettings detectDevice=true />
```
If the detectDevice attribute is set to true, ColdFusion automatically detects the features and capabilities of the device (width, height, and orientation) on which the application is running.

The following example shows the usage of the device detection feature:

```coldfusion
<cfClientSettings detectDevice=true />

<cfclient>

<cffunction access="public" name="showCanvasSupport" returntype="void">
    <cfset evalStr = "document.getElementById('canvas').innerHTML='"
    & cfclient.canvas & "'">
</cffunction>

</cfclient>

Canvas support -<b id="canvas"></b><br>
<button onclick="invokeCFClientFunction('showCanvasSupport',null)"
   >Show canvas support</button>
```

In the above example, we are trying to find if the device supports HTML5 Canvas. `cfclient.canvas` returns a boolean value indicating the support for the HTML5 Canvas property.

ColdFusion Server internally uses [Modernizer JavaScript library](https://github.com/Modernizr/Modernizr) (version 2.6.2) for the device detection feature.

The following table lists the supported device features with example usage:

<table>
<thead>
<tr>
<th>Features</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch Events</td>
<td>cfclient.touch</td>
</tr>
<tr>
<td>Canvas Text</td>
<td>cfclient.canvastext</td>
</tr>
<tr>
<td>Canvas</td>
<td>cfclient.canvas</td>
</tr>
<tr>
<td>Geolocation</td>
<td>cfclient.geolocation</td>
</tr>
<tr>
<td>Web Sockets</td>
<td>cfclient.websockets</td>
</tr>
<tr>
<td>Drag ‘n Drop</td>
<td>cfclient.draganddrop</td>
</tr>
<tr>
<td>History</td>
<td>cfclient.history</td>
</tr>
<tr>
<td>Feature</td>
<td>JavaScript Method</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>applicationCache</td>
<td>cfclient.applicationcache</td>
</tr>
<tr>
<td>localStorage</td>
<td>cfclient.localstorage</td>
</tr>
<tr>
<td>Width</td>
<td>cfclient.width</td>
</tr>
<tr>
<td>Height</td>
<td>cfclient.height</td>
</tr>
<tr>
<td>Device Width</td>
<td>cfclient.deviceWidth</td>
</tr>
<tr>
<td>Device Height</td>
<td>cfclient.deviceHeight</td>
</tr>
<tr>
<td>Orientation</td>
<td>cfclient.orientation</td>
</tr>
<tr>
<td>Device Group Name</td>
<td>cfclient.deviceGroupName</td>
</tr>
<tr>
<td>Device Group Descriptions</td>
<td>cfclient.deviceGroupDescription</td>
</tr>
<tr>
<td>CSS Animations</td>
<td>cfclient.cssanimations</td>
</tr>
<tr>
<td>CSS Columns</td>
<td>cfclient.csscolumns</td>
</tr>
<tr>
<td>CSS Generated Content</td>
<td>cfclient.generatedcontent</td>
</tr>
<tr>
<td>CSS Gradients</td>
<td>cfclient.cssgradients</td>
</tr>
<tr>
<td>CSS Reflections</td>
<td>cfclient.cssreflections</td>
</tr>
<tr>
<td>CSS 2D Transforms</td>
<td>cfclient.csstransforms</td>
</tr>
<tr>
<td>CSS 3D Transforms</td>
<td>cfclient.csstransforms3d</td>
</tr>
<tr>
<td>CSS Transitions</td>
<td>cfclient.csstransitions</td>
</tr>
<tr>
<td>Audio</td>
<td>cfclient.audio</td>
</tr>
<tr>
<td>Video</td>
<td>cfclient.video</td>
</tr>
<tr>
<td>Hash Change</td>
<td>cfclient.hashchange</td>
</tr>
<tr>
<td>IndexedDB</td>
<td>cfclient.indexeddb</td>
</tr>
<tr>
<td>Input Attributes</td>
<td>cfclient.input.* (refers to attributes for input elements. For possible values, see the Modernizr documentation)</td>
</tr>
<tr>
<td>Input Types</td>
<td>cfclient.inputtypes.* (refers to input type attributes. For possible values, see the Modernizr documentation)</td>
</tr>
</tbody>
</table>
For the description on all above mentioned features, see the [Modernizr documentation](#).

### Using media queries

Media queries allow you to apply changes to the page design based on the viewing size and capability of the device on which your content is displayed. A media query consists of one or more logical expressions formed using the detected device data that checks for certain conditions of media feature and based on the result of this expression we can change the layout of the page dynamically.

If you are building a mobile application, you can easily detect the characteristics of the device and customize the layout just for that device as shown in the following example:

```cfml
<cfclientsettings detectDevice=true />
<cfclient>
  <cfif cfclient.width lte 480 >
    <cfinclude template=" phone.css ">
  <cfelseif cfclient.width gte 480 AND cfclient.width lte 760>
    <cfinclude template=" tablet.css ">
  <cfelse>
    <cfinclude template=" desktop.css ">
  </cfif>
</cfclient>
```

In the above example, the web page is customized for different devices based on their screen sizes.

### Handling orientation changes

For handling the device orientation changes, you can register a listener using the `addOrientationListener()` function:
In the above example, addOrientationListener function is used to register a listener that monitors the orientation of the device (landscape or portrait). When the orientation of the device changes, an orientationHandler call back function is invoked.

You can use the removeOrientationListener to un-register the listener:

```coldfusion
<cffunction access="public" name="removeorientationhandler" returntype="void">
    <cfset cfclient.removeOrientationListener(orientationhandler)>
</cffunction>
```
You can also add multiple listeners:

<cfset cfclient.addOrientationListener(orientationHandler1)>  
<cfset cfclient.addOrientationListener(orientationHandler2)>

When the device orientation changes, all the registered listener functions are invoked.

Handling window resizing events

For handling the window resizing events, you can register a listener using the addResizeListener() function:

<cfclientsettings detectDevice=true />  
<cfclient>
  <cfoutput>
    Width :<b id="width"></b><br>
    Height :<b id="height"></b><br>
    Device width :<b id="devicewidth"></b><br>
    Device height :<b id="deviceheight"></b><br>
  </cfoutput>
  <!--- Adding the resize handler here. After adding the handler, the handler will be invoked whenever there is a browser resize. --->
  <cfset cfclient.addResizeListener(resizehandler)>
  <cffunction access="public" name="resizehandler" returntype="void" >
    <cfargument name="width" type="string">
    <cfargument name="height" type="string">
    <!--- The width and height of the browser will be passed as arguments to the handler. You can also get the width/height value from cfclient. --->
    <cfset evalstr = "document.getElementById('width').innerHTML='' & #width# & ''">
    <cfset evalstr = "document.getElementById('height').innerHTML='' & #height# & ''">
    <cfset evalstr = "document.getElementById('devicewidth').innerHTML='' & #cfclient.deviceWidth# & ''">
    <cfset evalstr = "document.getElementById('deviceheight').innerHTML='' & #cfclient.deviceHeight# & ''">
  </cffunction>
</cfclient>
You can also add multiple listeners:

```coldfusion
<cfset cfclient.addResizeListener(resizeHandler1)>
<cfset cfclient.addResizeListener(resizeHandler2)>
```

When there is a change in window size all the registered resize listener functions are invoked. You can use removeResizeListener() to un-register the handlers.

```coldfusion
<cffunction access="public" name="removeresizehandler"
    returntype="void" >
    <cfset cfclient.removeResizeListener(resizeHandler)>
</cffunction>
```

**Setting device timeout**

In the `<cfclientsettings>` tag, an attribute called deviceTimeOut can be specified. The default value of deviceTimeout is 10 secs. When enableDeviceApi or detectDevice is set as true, the deviceTimeOut value will be honored. Time will be provided for the required plugins to be loaded. After the specified time, an exception will be thrown.

```coldfusion
<cfclientsettings detectDevice=true deviceTimeOut="30" />
Tags d-e

This section lists tags starting with alphabets D-E with their description.

- cfdbinfo
- cfdefaultcase
- cfdirectory
- cfdiv
- cfdocument
- cfdocumentitem
- cfdocumentsection
- cfdump
- cfelse
- cfelseif
- cferror
- cfexchangecalendar
- cfexchangecollection
- cfexchangeconnection
- cfexchangecontact
- cfexchangeconversation
- cfexchangefilter
- cfexchangefolder
- cfexchangemail
- cfexchangetask
- cfexecute
- cfexit
**cfdbinf**

**Description**

 Lets you retrieve information about a data source, including details about the database, tables, queries, procedures, foreign keys, indexes, and version information about the database, driver, and JDBC.

**Category**

*Database manipulation tags*

**Syntax**

```xml
<cfdbinf
datasource="data source name"
name="result name"
type="dbnames|tables|columns|version|procedures|foreignkeys|index"
dbname="database name"
password="password"
pattern="filter pattern"
table="table name"
username="username">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

*cfinset, cprocparam, cfprocresult, cfqueryparam, cfstoredproc, cftnaction, cfupdate; Opt
imizing database use* in the *Developing ColdFusion Applications*.

**History**

ColdFusion 8: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>Optional</td>
<td></td>
<td>Datasource to use to connect to the database.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Name to use to refer to the result.</td>
</tr>
<tr>
<td>type</td>
<td>Required</td>
<td>Type of information to get:</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>dbnames: database name and type</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>tables: name, type, and remarks</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>columns: name, SQL data type, size, decimal precision, default value, maximum length in bytes of a character or integer data type column, whether nulls are allowed, ordinal position, remarks, whether the column is a primary key, whether the column is a foreign key, the table that the foreign key refers to, the key name the foreign key refers to</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>version: database product name and version, driver name and version, JDBC major and minor version</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>procedures: name, type, and remarks</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>foreignkeys: foreign key name and table, primary key name, delete, and update rules</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>index: name, column on which the index is applied, ordinal position, cardinality, whether the row represents a table statistic or an index, number of pages used by the table or index, whether the index values are unique</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ClientInfo: The client information metadata for the specified data source</td>
<td></td>
</tr>
</tbody>
</table>
### Usage

Use the `cfdbinfo` tag to return a query object that contains information about a database. The query object varies, depending on the value that you specify in the `type` attribute. The following table lists the query object contents for each type:

<table>
<thead>
<tr>
<th>Type</th>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dbnames</code></td>
<td>DATABASE_NAME</td>
<td>Name of the database.</td>
</tr>
<tr>
<td></td>
<td>TYPE</td>
<td>Type of the database, whether schema or catalog.</td>
</tr>
<tr>
<td><code>tables</code></td>
<td>TABLE_NAME</td>
<td>Name of the table.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>REMARKS</td>
<td>Remarks of the table.</td>
<td></td>
</tr>
<tr>
<td>columns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>Name of the column.</td>
<td></td>
</tr>
<tr>
<td>TYPE_NAME</td>
<td>SQL data type of the column.</td>
<td></td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>Whether the column allows nulls.</td>
<td></td>
</tr>
<tr>
<td>IS_PRIMARYKEY</td>
<td>Whether the column is a primary key.</td>
<td></td>
</tr>
<tr>
<td>IS_FOREIGNKEY</td>
<td>Whether the column is a foreign key.</td>
<td></td>
</tr>
<tr>
<td>REFERENCED_PRIMARYKEY</td>
<td>If the column is a foreign key, the name of the table it refers to.</td>
<td></td>
</tr>
<tr>
<td>REFERENCED_PRIMARYKEY_TABLE</td>
<td>If the column is a foreign key, the key name it refers to.</td>
<td></td>
</tr>
<tr>
<td>COLUMN_SIZE</td>
<td>Size of the column</td>
<td></td>
</tr>
<tr>
<td>DECIMAL_DIGITS</td>
<td>Number of digits to the right of the decimal point.</td>
<td></td>
</tr>
<tr>
<td>COLUMN_DEFAULT_VALUE</td>
<td>Default value of column.</td>
<td></td>
</tr>
<tr>
<td>CHAR_OCTET_LENGTH</td>
<td>Maximum length in bytes of a character or integer data type column.</td>
<td></td>
</tr>
<tr>
<td>ORDINAL_POSITION</td>
<td>Ordinal position of the column.</td>
<td></td>
</tr>
<tr>
<td>REMARKS</td>
<td>Remarks of the column.</td>
<td></td>
</tr>
</tbody>
</table>

<p>| version    |                                                                            |
| DATABASE_VERSION| Version of the database management system.       |
| DATABASE_PRODUCTNAME| Name of the database management system.             |
| DRIVER_VERSION| Version of the database driver.                  |
| DRIVER_NAME  | Name of the database driver.                  |</p>
<table>
<thead>
<tr>
<th>PROCEDURE_NAME</th>
<th>Major version number of the driver.</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC_MINOR_VERSION</td>
<td>Minor version number of the driver.</td>
</tr>
<tr>
<td>procedures</td>
<td>Name of the stored procedure.</td>
</tr>
<tr>
<td>REMARKS</td>
<td>Remarks for the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_TYPE</td>
<td>Procedure type, which indicates whether the procedure returns a result.</td>
</tr>
<tr>
<td>foreignkeys</td>
<td>Foreign key name.</td>
</tr>
<tr>
<td>FKCOLUMN_NAME</td>
<td>Foreign key table name.</td>
</tr>
<tr>
<td>FKTABLE_NAME</td>
<td>Primary key name.</td>
</tr>
<tr>
<td>PKCOLUMN_NAME</td>
<td>Specifies what action to take when you delete a record that has dependent records.</td>
</tr>
<tr>
<td>DELETE_RULE</td>
<td>Specifies what action to take when you update a record that has dependent records.</td>
</tr>
<tr>
<td>UPDATE_RULE</td>
<td>Name of the index, empty if type is table statistic.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>Name of the column on which the index is applied, empty if the type is table statistic.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>Ordinal position.</td>
</tr>
<tr>
<td>ORDINAL_POSITION</td>
<td>Number of unique values if the type is index, or number of rows if the type is statistic.</td>
</tr>
<tr>
<td>CARDINALITY</td>
<td>Whether the row represents a table statistic or an index. Index types are clustered, hashed, or other.</td>
</tr>
<tr>
<td>TYPE</td>
<td>Number of pages used by the table if the type is table statistic, or the number of pages used by the index.</td>
</tr>
<tr>
<td>PAGES</td>
<td></td>
</tr>
</tbody>
</table>
Whether the index values are unique.

### Example

```coldfusion
<cfset datasrc = "oratest">
<cfdbinfo type="dbnames" datasource="#datasrc#" name="dbdata">
<cfoutput>
The #datasrc# data source has the following databases:<br/>
</cfoutput>
<table border="1">
<tr>
<th valign="top" align="left">Database name</th><th>Type</th>
</tr>
<cfoutput query="dbdata">
<tr>
<td>#dbdata.DATABASE_NAME#</td><td>#dbdata.TYPE#</td>
</tr>
</cfoutput>
</table>
```
**cfdefaultcase**

**Description**

Used only inside the `cfswitch` tag body. Contains code to execute when the expression specified in the `cfswitch` tag does not match the value specified by a `cfcase` tag.

**Category**

**Flow-control tags**

**Syntax**

```cfcfdefaultcase>
</cfdefaultcase>```

**See also**

`cfcase`, `cfswitch`; `cfswitch`, `cfcase`, and `cfdefaultcase` in the *Developing ColdFusion Applications*

**History**

ColdFusion MX: Changed placement requirements: this tag does not have to follow all `cfcase` tags in the `cfswitch` tag body.

**Usage**

The contents of the `cfdefaultcase` tag body executes if the expression attribute of the `cfswitch` tag does not match any of the values specified by the `cfcase` tags in the `cfswitch` tag body. The contents of the `cfdefaultcase` tag body can include HTML and text, and CFML tags, functions, variables, and expressions.

You can specify only one `cfdefaultcase` tag within a `cfswitch` tag. You can put the `cfdefaultcase` tag at any position within a `cfswitch` statement; it is not required to be the last item, but it is good programming practice to put it last.

**Example**
<!-- The following example displays a grade based on a 1-10 score. Several of the cfcase tags match more than one score. For simplicity, the example sets the score to 7. -->
<cfset score="7">
<cfswitch expression="#score#">
<cfcase value="10">
    <cfset grade="A">
</cfcase>
<cfcase value="9;8" delimiters=";"">
    <cfset grade="B">
</cfcase>
<cfcase value="7;6" delimiters=";"">
    <cfset grade="C">
</cfcase>
<cfcase value="5;4;" delimiters=";"">
    <cfset grade="D">
</cfcase>
<cfdefaultcase>
    <cfset grade="F">
</cfdefaultcase>
</cfswitch>
<cfoutput>
Your grade is #grade#
</cfoutput>
cfdirectory

Description

Manages interactions with directories.

Category

File management tags

Syntax

```cfdirectory
directory = "directory name"
action = "list|copy|create|delete|rename"
destination = "full pathname"
filter = "list filter"
listInfo = "name|all"
mode = "permission"
name = "query name"
newDirectory = "new directory name"
recurse = "yes|no"
sort = "sort specification"
storeACL = "S3_premissions"
storeLocation = "location"
type = "file|dir|all">
```

⚠️ Note

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

See also

cffile

History

ColdFusion 10: Added the action `copy` and the attribute `destination`.

ColdFusion 9.0.1: Added the `storeACL` and `storeLocation` attributes.

ColdFusion 8: Added the `listinfo` and `type` attributes.

ColdFusion MX 7: Added the `recurse` attribute and `directory` result set column.

ColdFusion MX:

1. Changed behavior for `action = "list"`:
   - On Windows, `cfdirectory action = "list"` no longer returns the directory entries "." (dot) or ".." (dot dot), which represent "the current directory" and "the parent directory."
   - On Windows, `cfdirectory action = "list"` no longer returns the values of the Archive and System attributes.
• On UNIX and Linux, \texttt{cfdirectory action = "list"} does not return any information in the \texttt{mode column}.

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory</td>
<td>Required</td>
<td></td>
<td>Absolute pathname of directory against which to perform action. You can use an IP address, as in the following example:</td>
</tr>
<tr>
<td>action</td>
<td>Optional</td>
<td>list</td>
<td>list: returns a query record set of the files in the specified directory. The directory entries &quot;.&quot; (dot) and &quot;..&quot; (dot dot), which represent the current directory and the parent directory, are not returned.</td>
</tr>
<tr>
<td>destination</td>
<td>Required if \texttt{action = &quot;copy&quot;}</td>
<td></td>
<td>Path of the destination directory. If not an absolute path, it is relative to the source directory.</td>
</tr>
<tr>
<td>filter</td>
<td>Optional if \texttt{action = &quot;list&quot;}</td>
<td></td>
<td>File extension filter applied to returned names, for example, &quot;.\texttt{.cfm}&quot;. One filter can be applied.</td>
</tr>
</tbody>
</table>
| listinfo   | Optional | all       | - all: includes all information in the result set.  
|           |          |          | - name: includes only filenames in the result set. |
| mode      | Optional |           | Used with action = "create". Permissions. Applies only to UNIX and Linux. Octal values of chmod command. Assigned to owner, group, and other, respectively, for example:  
|           |          |           | - 644: assigns read/write permission to owner; read permission to group and other.  
|           |          |           | - 777: assigns read/write/execute permission to all. |
| name      | Required if action = "list" | Name for output record set. |
| newDirectory | Required if action = "rename" | New name for directory. |
| recurse   | Optional | no        | Whether ColdFusion performs the action on subdirectories:  
|           |          |           | - yes  
|           |          |           | - no  
|           |          |           | Valid for action="list" and action="delete". |
| **sort** | Optional; used if `action = "list"` | **ASC** | Query columns by which to sort a directory listing. Delimited list of columns from query output. To qualify a column, use one of the following values:

- `asc`: ascending (a to z) sort order.
- `desc`: descending (z to a) sort order.

For example:
```
sort = "directory ASC, size DESC, datelast modified "
```

| **storeACL** | Optional; used if `action = "create"` | | An array of struct where each struct represents a permission or grant. For details, see Using Amazon S3 storage in Optimizing ColdFusion applications.

| **StoreLocation** | Optional; used if `action = "create"` | **US** | Used to change the location of the created bucket. The location can either be EU, US, or US-WEST. For details, see Using Amazon S3 storage in Optimizing ColdFusion applications.

| **type** | Optional | **all** | • file: includes only filenames.
• dir: includes only directory names.
• all: includes both filenames and directory names. |
storeLocation  |  Optional  |  Used to change the location of the created bucket. The location can either be EU or US. The default location is US.

storeACL     |  Optional  |  An array of struct where each struct represents a permission or grant.

Usage

If you put ColdFusion applications on a server that is used by multiple customers, you must consider the security of files and directories that could be uploaded or otherwise manipulated with this tag by unauthorized users. For more information about securing ColdFusion tags, see Configuring and Administering ColdFusion.

If `action = "list"`, `cfdirectory` returns the following result columns, which you can reference in a `cfoutput` tag:

- **name**: Directory entry name. The entries "." and ". ." are not returned.
- **directory**: Directory that contains the entry.
- **size**: Directory entry size.
- **type**: File type: `file`, for a file; `dir`, for a directory.
- **dateLastModified**: The date that an entry was last modified.
- **attributes**: File attributes, if applicable.
- **mode**: Empty column; retained for backward compatibility with ColdFusion 5 applications on UNIX.

Use the following result columns in standard CFML expressions, preceding the result column name with the query name:

```coldfusion
#mydirectory.name#
#mydirectory.directory#
#mydirectory.size#
#mydirectory.type#
#mydirectory.dateLastModified#
#mydirectory.attributes#
#mydirectory.mode#
```

Note

If the `cfdirectory` tag does not appear to work, for example, if a list operation returns an empty result set, make sure that you have correct permissions to access the directory. For example, if you run ColdFusion as a service on Windows, it operates by default as System, and cannot access directories on a remote system or mapped drive; to resolve this issue, do not run ColdFusion using the local system account.

The `filter` attribute specifies a pattern of one or more characters. All names that match that pattern are included in the list. On Windows systems, pattern matching ignores text case, on UNIX and Linux, pattern matches are case-sensitive. The following two characters have special meaning in the pattern and are called metacharacters:

- The asterisk (*) matches any zero or more characters.
The question mark (?) matches any single character. The following table shows examples of patterns and filenames that they match:

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo.*</td>
<td>Any file called foo with any extension; for example, foo.html, foo.cfm, and foo.xml.</td>
</tr>
<tr>
<td>*.html</td>
<td>All files with the suffix .html, but not files with the suffix .htm.</td>
</tr>
<tr>
<td>??</td>
<td>All files with two-character names.</td>
</tr>
</tbody>
</table>

Example

```cfml
<!---- EXAMPLE 1: Creating and Renaming
Check that the directory exists to avoid getting a ColdFusion error message. ---->
<cfset newDirectory = "otherNewDir">
<cfset currentDirectory = GetDirectoryFromPath(GetCurrentTemplatePath()) & "newDir">

<!---- Check whether the directory exists. ---->
<cfif DirectoryExists(currentDirectory)>
  <!---- If yes, rename the directory. ---->
  <cfdirectory action = "rename" directory = "#currentDirectory#"
               newDirectory = "#newDirectory#" >
  <cfoutput>
    <p>The directory existed and the name has been changed to: #newDirectory#</p>
  </cfoutput>
</cfif>
<!---- If no, create the directory. ---->
<cfelse>
  <!---- If no, create the directory. ---->
  <cfdirectory action = "create" directory = "#currentDirectory#" >
  <cfoutput>
    <p>Your directory has been created.</p>
  </cfoutput>
</cfif>

<!---- EXAMPLE 2: Deleting a directory
Check that the directory exists and that files are not in the directory to avoid getting ColdFusion error messages. ---->

<cfset currentDirectory = GetDirectoryFromPath(GetCurrentTemplatePath()) & "otherNewDir">
<!---- Check whether the directory exists. ---->
<cfif DirectoryExists(currentDirectory)>
  <!---- If yes, check whether there are files in the directory before deleting. ---->
  <cfdirectory action="list" directory="#currentDirectory#"
              name="myDirectory">
    <cfif myDirectory.recordcount gt 0>
      <!---- If yes, delete the files from the directory. ---->
      <cfoutput>
        <p>Files exist in this directory. Either delete the files or code something to do so.</p>
      </cfoutput>
    </cfif>
  </cfif>
<!---- Directory is empty - just delete the directory. ---->
<cfelse>
  <!---- Directory is empty - just delete the directory. ---->
  <cfdirectory action = "delete" directory = "#currentDirectory#">
  <cfoutput>
```
```
<p>The directory existed and has been deleted.</p>

<cfif>
</cfif>
<cfelse>
<!--- If no, post message or do some other function. --->
<cfoutput><p>The directory did NOT exist.</p></cfoutput>
</cfif>

<!---EXAMPLE 3: List directories
The following example creates both an array of directory names and a query that
contains entries for the directories only. --->

<cfdirectory directory="C:/temp" name="dirQuery" action="LIST">

<!--- Get an array of directory names. --->
<cfset dirsArray=arraynew(1)>
<cfset i=1>
<cfloop query="dirQuery">
<cfif dirQuery.type IS "dir">
    <cfset dirsArray[i]=dirQuery.name>
    <cfset i = i + 1>
</cfif>
</cfloop>
<cfdump var="#dirsArray#">
<br>

<!--- Get all directory information in a query of queries. --->
<cfquery dbtype="query" name="dirsOnly">
SELECT * FROM dirQuery
WHERE TYPE='Dir'
Another example

The following example was contributed by Steve Blades:

```cfc
component {
    VARIABLES.instance.rootPath = "H:\Karaoke Library\";
    /**
     * @access remote
     * @returnType struct
     * @output false
     */
    function GetSubordinates (string folder="", string parentage="") {
        var retVal = {
            'success':true,
            'message':'',
            'data':''
        };
        try {
            // script version of cfdirectory
            var dir = DirectoryList(VARIABLES.instance.rootPath &
                                    ARGUMENTS.parentage & ARGUMENTS.folder, false, "query");
            LOCAL.retVal.data = LOCAL.dir;
        } catch (any err) {
            LOCAL.retVal.success = false;
            LOCAL.retVal.message = err.message;
            LOCAL.retVal['errorCode'] = err.errorcode;
        }
        return LOCAL.retVal;
    }
}
```
**cfdiv**

**Description**

Creates an HTML div tag or other HTML container tag and lets you use asynchronous form submission or a bind expression to dynamically control the tag contents.

**Category**

*Display management tags*

**Syntax**

```
<cfdiv
  bind = "bind expression"
  bindOnLoad = "true|false"
  ID = "HTML tag ID"
  onBindError = "JavaScript function name"
  tagName = "HTML tag name"
/>
```

OR

```
<cfdiv
  ID = "HTML tag ID"
  tagName = "HTML tag name">
  tag body contents
</cfdiv>
```

If the tag does not have a body and end tag, close it with /> character combination.

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute name as structure key.

**See also**

*cfajaximport, cflayout, cfpod, cfwindow*

**History**

ColdFusion 8: Added this tag

**Attributes**

The following table lists attributes that ColdFusion uses directly. The tag passes any other attributes that you specify directly as tag attributes to the generated HTML tag.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
| bind      | Optional | A bind expression that returns the container contents. If you specify this attribute the `<cfdiv>` tag cannot have a body. **Note:** If a CFML page specified in this attribute contains tags that use AJAX features, such as `<cfgrid>`, `<cfform>`, and `<cfwindow>`, you must use a `<cfajaximport>` tag on the page with the `<cfdiv>` tag. For more information, see `<cfajaximport>`.

<table>
<thead>
<tr>
<th>bindOnLoad</th>
<th>Optional</th>
<th>true</th>
</tr>
</thead>
</table>
|            |          | • *true:* executes the `bind` attribute expression when first loading the tag.  
|            |          | • *false:* does not execute the `bind` attribute expression until the first bound event. To use this attribute, also specify a `bind` attribute. For more information, see *Using the bindOnLoad attribute* in *Using Ajax User Interface Components and Features* in the *Developing ColdFusion Applications*.

| ID         | Optional | The HTML ID attribute value to assign to the generated container tag. |
onBindError | Optional | See Description | The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and have specified a global error handler (by using the ColdFusion.setGlobalErrorHandler function), it displays the error message; otherwise a default error pop-up window appears. To use this attribute, also specify a bind attribute.

tagName | Optional | DIV | The HTML container tag to create.

Usage

By default, the cfdiv tag creates a div HTML element. You can use standard HTML and CSS techniques to control the position and appearance of the element and its contents. Use the tagName attribute to create and populate an HTML content element, such as span or b. Use the cfdiv tag to create tags that can take HTML markup content directly in the body, such as span, i, b, or p, and not for tags that cannot, such as input, option, and frameset.

If you submit a form that is inside a cfdiv tag (including in HTML returned by a bind expression), the form submits asynchronously, and the response from the form submission populates the cfdiv region.

If you specify a bind attribute, the tag dynamically populates the element using a bind expression. The bind expression can specify a CFC function, a JavaScript function, a URL, or a string that contains bind parameters. An animated icon and the text "Loading..." appears while the contents are being fetched. For detailed information on using the bind attribute and bind expressions, see Using Ajax Data and Development Features in the Developing ColdFusion Applications.

Example

The following simple example shows how you can use the cfdiv tag. It uses binding to display the contents of a text input field in an HTML DIV region.

The cfdivtag.cfm file, the main application file, has the following contents.
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>cfdiv Example</title>
</head>
<body>
<cfform>
  <cfinput name="tinput1" type="text">
</cfform>

<h3> using a div</h3>
<cfdiv bind="url:divsource.cfm?InputText={tinput1}" ID="theDiv"
  style="background-color:##CCffFF; color:red; height:350"/>
</body>
</html>

The divsource.cfm file that defines the contents of the div region has the following code:

```coldfusion
<h3>Echoing main page input:</h3>
<cfoutput>
  <cfif isdefined("url.InputText") AND url.InputText NEQ ">">
    #url.InputText#
  <cfelse>
    No input
  </cfif>
</cfoutput>
```

To test the code, run the cfdivtag.cfm page, enter some text, and tab out of the text box or click outside the text box. The div region appears with a light blue background and red text, and when you exit the text box, it shows the text you entered.
cfdocument

Description

Creates PDF or FlashPaper output from a text block containing CFML and HTML.

Category

Data output tags

Syntax

```
<cfdocument
    format = "PDF|FlashPaper"
    authPassword = "authentication password"
    authUser = "authentication user name"
    backgroundVisible = "yes|no"
    bookmark = "yes|no"
    encryption = "128-bit|40-bit|none"
    filename = "filename"
    fontEmbed = "yes|no"
    formfields = "yes|no"
    formsType = "FDF|PDF|HTML|XML"
    localUrl = "yes|no"
    marginBottom = "number"
    marginLeft = "number"
    marginRight = "number"
    marginTop = "number"
    mimeType = "text/plain|application/xml|image/jpeg|image/png|image/bmp|image/gif"
    name = "output variable name"
    openPassword = "password to open protected documents"
    orientation = "portrait|landscape"
    overwrite = "yes|no"
    ownerPassword = "password"
    pageHeight = "page height in inches"
    pageType = "page type"
    pageWidth = "page width in inches"
    pdfa = "yes|no"
    permissions = "permission list"
    permissionsPassword = "password to access restricted permissions"
    proxyHost = "IP address or server name for proxy host"
    proxyPassword = "password for the proxy host"
    proxyPort = "port of the proxy host"
    proxyUser = "user name for the proxy host"
    saveAsName = "PDF filename"
    scale = "percentage less than 100"
    src = "URL|pathname relative to web root"
    srcfile = "absolute pathname to a file"
    tagged = "yes|no"
    unit = "in|cm"
    userAgent = "HTTP user agent identifier"
    userPassword = "password">
    HTML and CFML code
</cfdocument>
```
Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also
cfdocumentitem, cfdocumentsection, cfform, cfpdf, cfpdfform, cfpresentation, cfprint, cfreport

History

ColdFusion 9: Add ppt support to the srcFile attribute. Added the following attributes to support conversion of a Word document to PDF or HTML using OpenOffice libraries:

- formfields attribute
- formsType attribute
- openpassword attribute
- permissionspassword attribute
- pdfa attribute
- tagged attribute

ColdFusion 8: Added the following attributes and variables:

- bookmark attribute
- localUrl attribute
- Ability to embed existing PDF forms by using the cfpdfform tag in the cfdocument tag.
- ColdFusion determines the MIME type of a source file based on the source filename, if the mimeType attribute is not specified.
- Ability to pass a PDF variable created with the cfdocument tag as the source for the cfpdf tag.
- authPassword, authUser, proxyHost, proxyPassword, proxyPort, proxyUser, and userAgent attributes
- saveAsName attribute
- totalsectionpagecount and currentsectionpagenumber scope variables.

ColdFusion MX 7.01: Added the src, srcfile, and mimetype attributes.

ColdFusion MX 7: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authPassword</td>
<td>Optional</td>
<td></td>
<td>Password sent to the target URL for Basic Authentication. Combined with <code>username</code> to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.</td>
</tr>
<tr>
<td>authUser</td>
<td>Optional</td>
<td></td>
<td>User name sent to the target URL for Basic Authentication. Combined with <code>password</code> to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication.</td>
</tr>
</tbody>
</table>
| backgroundVisible | Optional | no | Specifies whether the background prints when the user prints the document:  
  - yes: includes the background when printing.  
  - no: does not include the background when printing. |
| bookmark | Optional | no | Specifies whether bookmarks are created in the document:  
  - yes: creates bookmarks.  
  - no: does not create bookmarks. |
| encryption | Optional | none | (format="PDF" only) Specifies whether the output is encrypted:  
  - 128-bit  
  - 40-bit  
  - none |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Optional</td>
<td></td>
<td>Pathname of a file to contain the PDF or FlashPaper output. If you omit the <code>filename</code> attribute, ColdFusion displays the output in the browser.</td>
</tr>
<tr>
<td>fontEmbed</td>
<td>Optional</td>
<td>yes</td>
<td>Specifies whether ColdFusion embeds fonts in the output:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes: embeds fonts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no: does not embed fonts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- selective: embeds all fonts except Java fonts and core fonts.</td>
</tr>
<tr>
<td>format</td>
<td>Required</td>
<td></td>
<td>Report format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- FlashPaper</td>
</tr>
<tr>
<td>formfields</td>
<td>Optional</td>
<td>yes</td>
<td>This attribute is available only if you have integrated OpenOffice with ColdFusion. A Boolean value that specifies if form fields are exported as widgets or only their fixed print representation is exported.</td>
</tr>
<tr>
<td>formstype</td>
<td>Optional</td>
<td>FDF</td>
<td>This attribute is available only if you have integrated OpenOffice with ColdFusion. Specifies the submitted format of a PDF form. It can be one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- HTML</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- XML</td>
</tr>
</tbody>
</table>
| localUrl    | Optional | no       | Specifies whether to retrieve image files directly from the local drive:  
|            |          |         |  
|            |          |         | - yes: ColdFusion retrieves image files directly from the local drive rather than by using HTTP, HTTPS, or proxy.  
|            |          |         | - no: ColdFusion uses HTTP, HTTPS, or proxy to retrieve image files even if the files are stored locally. For more information, see the "Using an image file URL" section.  
| marginBottom | Optional |         | Bottom margin in inches (default) or centimeters. To specify the bottom margin in centimeters, include the `unit=cm` attribute.  
| marginLeft  | Optional |         | Left margin in inches (default) or centimeters. To specify the left margin in centimeters, include the `unit=cm` attribute.  
| marginRight | Optional |         | Right margin in inches (default) or centimeters. To specify the right margin in centimeters, include the `unit=cm` attribute.  
| marginTop   | Optional |         | Top margin in inches (default) or centimeters. To specify the top margin in centimeters, include the `unit=cm` attribute.  
|            |          |         |  

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mimeType</td>
<td>Optional</td>
<td>text/html</td>
<td>MIME type of the source document. Supported MIME types are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- text/html</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- text/plain</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- application/xml</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- image/bmp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- image/jpeg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- image/png</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- image/gif</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If you do not specify this attribute explicitly, ColdFusion uses the filename to determine the MIME type.</td>
</tr>
<tr>
<td>name</td>
<td>Optional</td>
<td></td>
<td>Name of an existing variable into which the tag stores the PDF or FlashPaper output.</td>
</tr>
<tr>
<td>openpassword</td>
<td>Optional</td>
<td></td>
<td>This attribute is available only if you have integrated OpenOffice with ColdFusion.Password required to open a password-protected document.</td>
</tr>
<tr>
<td>orientation</td>
<td>Optional</td>
<td>portrait</td>
<td>Page orientation:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- portrait</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- landscape</td>
</tr>
<tr>
<td>overwrite</td>
<td>Optional</td>
<td>no</td>
<td>Specifies whether ColdFusion overwrites an existing file. Used in conjunction with the file name attribute.</td>
</tr>
<tr>
<td>ownerPassword</td>
<td>Optional</td>
<td></td>
<td>(format=&quot;PDF&quot; only) Specify the owner password. Cannot be same as userPassword.</td>
</tr>
<tr>
<td>attribute</td>
<td>type</td>
<td>value</td>
<td>description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pageHeight</td>
<td>Optional</td>
<td></td>
<td>Page height in inches (default) or centimeters. This attribute is only valid if pagetype=custom. To specify page height in centimeters, include the unit=cm attribute.</td>
</tr>
<tr>
<td>pageType</td>
<td>Optional</td>
<td>letter</td>
<td>Page type into which ColdFusion generates the report:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- legal: 8.5 inches x 14 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- letter: 8.5 inches x 11 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- A4: 8.27 inches x 11.69 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- A5: 5.81 inches x 8.25 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- B4: 9.88 inches x 13.88 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- B5: 7 inches x 9.88 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- B4-JIS: 10.13 inches x 14.31 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- B5-JIS: 7.19 inches x 10.13 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- custom: custom height and width. If you specify custom, also specify the pageHeight and pageWidth attributes, can optionally specify margin attributes and whether the units are inches or centimeters.</td>
</tr>
<tr>
<td>pageWidth</td>
<td>Optional</td>
<td></td>
<td>Page width in inches (default) or centimeters. This attribute is only valid if pagetype=custom. To specify page width in centimeters, include the unit=cm attribute.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>pdfa</td>
<td>Optional</td>
<td>no</td>
<td>This attribute is available only if you have integrated OpenOffice with ColdFusion. A Boolean value that specifies if you need to create a PDF of type PDF/A-1 (ISO 19005-1:2005).</td>
</tr>
<tr>
<td>permissionpassword</td>
<td>Optional</td>
<td></td>
<td>This attribute is available only if you have integrated OpenOffice with ColdFusion. Password required to access restricted permissions. The restricted permissions are specified using the permissions attribute.</td>
</tr>
</tbody>
</table>
| permissions | Optional |       | (format="PDF" only) Sets one or more of the following permissions:  
- AllowPrinting  
- AllowModifyContents  
- AllowCopy  
- AllowModifyAnnotations  
- AllowFillIn  
- AllowScreenReaders  
- AllowAssembly  
- AllowDegradedPrinting  
Separate multiple permissions with commas. |
<p>| proxyHost  | Optional |       | Host name or IP address of a proxy server to which to send the request. |
| proxyPassword | Optional |       | Password required by the proxy server. |
| proxyPort  | Optional | 80    | The port to connect to on the proxy server. |
| proxyUser  | Optional |       | User name to provide to the proxy server. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scale</td>
<td>Optional</td>
<td>Scale factor as a percentage. Use this option to reduce the size of the HTML output so that it fits on that paper. Specify a number less than 100.</td>
</tr>
<tr>
<td>saveAsName</td>
<td>Optional</td>
<td>(format=&quot;PDF&quot; only) The filename that appears in the SaveAs dialog when a user saves a PDF file written to the browser.</td>
</tr>
<tr>
<td>src</td>
<td>Optional</td>
<td>URL or the relative path to the web root. You cannot specify both the src and srcfile attributes. The file must be in a browser-writable format such as, HTML, HTM, BMP, PNG, and so on.</td>
</tr>
<tr>
<td>srcfile</td>
<td>Optional</td>
<td>Absolute path of a file that is on the server. You cannot specify both the src and srcfile attributes. The file must be a PPT file, a Word file, or be in a browser-writable format such as, HTML, HTM, BMP, PNG, and so on.</td>
</tr>
<tr>
<td>tagged</td>
<td>Optional</td>
<td>no This attribute is available only if you have integrated OpenOffice with ColdFusion. A Boolean value that determines if the PDF is created using the Tagged PDF tag.</td>
</tr>
</tbody>
</table>
| unit        | Optional | in Default unit for the pageHeight, pageWidth, and margin attributes:  
  - in: inches.  
  - cm: centimeters. |
### userAgent

**Optional**

ColdFusion

Text to put in the HTTP User-Agent request header field. Used to identify the request client software.

### userPassword

**Optional**

(format="PDF" only)

Specifies a user password. Cannot be same as ownerPassword.

---

**Usage**

Use the `cfdocument` tag to render HTML and CFML output into PDF or FlashPaper format. ColdFusion does not return HTML and CFML outside of the `<cfdocument> </cfdocument>` pair.

The `cfdocument` tag can render HTML that supports the following standards:

- HTML 4.01
- XML 1.0
- DOM Level 1 and 2
- CSS1 and CSS2 (For more information, see the “Supported CSS styles” section).

The `cfdocument` tag does not support the Internet Explorer-specific HTML generated by Microsoft Word. Use the following syntax in the `filename` attribute to specify an in-memory file, which is not written to disk. In-memory files speed processing of transient data.

```
ram:///filepath
```

The filepath can include directories, for example `ram:///petStore/tracking/ordersummary.pdf`. Create the directories in the path before you specify the file. For more information on using in-memory files, see *Working with in-memory files* in the *Developing ColdFusion Applications*.

You can use the `src`, `srcfile`, and `mimeType` attributes to create PDF or FlashPaper output from a specified file or URL. Use the `src` and `srcfile` attributes instead of using the `cfhttp` tag to display the result in the `cfdocument` tag. When you specify the `src` or `srcfile` attributes, do not include any other content inside the `cfdocument` tag: ColdFusion ignores the additional content. The PDF or FlashPaper document returned by the `cfdocument` tag overwrites any previous HTML in the input stream and ignores any HTML after the `<cfdocument>` tag. You cannot embed a `cfreport` tag in a `cfdocument` tag.

---

**Note**

If you notice that the header text is cropped in the `cfdocument` tag output, increase the value of the `marginTop` attribute.

---

**Supported CSS styles**

The `cfdocument` tag supports the following CSS styles:
<table>
<thead>
<tr>
<th>background</th>
<th>background-attachment</th>
<th>background-color</th>
<th>background-image</th>
</tr>
</thead>
<tbody>
<tr>
<td>background-position</td>
<td>background-repeat</td>
<td>border</td>
<td>border-bottom</td>
</tr>
<tr>
<td>border-bottom-color</td>
<td>border-bottom-style (solid border only)</td>
<td>border-bottom-width</td>
<td>border-color</td>
</tr>
<tr>
<td>border-left</td>
<td>border-left-color</td>
<td>border-left-style (solid border only)</td>
<td>border-left-width</td>
</tr>
<tr>
<td>border-right</td>
<td>border-right-color</td>
<td>border-right-style (solid border only)</td>
<td>border-right-width</td>
</tr>
<tr>
<td>border-spacing</td>
<td>border-style (solid border only)</td>
<td>border-top</td>
<td>border-top-color</td>
</tr>
<tr>
<td>border-top-style (solid border only)</td>
<td>border-top-width</td>
<td>border-width</td>
<td>bottom</td>
</tr>
<tr>
<td>clear</td>
<td>clip</td>
<td>color</td>
<td>content (strings, counters only)</td>
</tr>
<tr>
<td>counter-increment</td>
<td>counter-reset</td>
<td>cursor</td>
<td>display</td>
</tr>
<tr>
<td>float</td>
<td>font</td>
<td>font-family</td>
<td>font-size</td>
</tr>
<tr>
<td>font-style</td>
<td>font-weight</td>
<td>height</td>
<td>left</td>
</tr>
<tr>
<td>letter-spacing</td>
<td>line-height</td>
<td>list-style-type</td>
<td>margin</td>
</tr>
<tr>
<td>margin-bottom</td>
<td>margin-left</td>
<td>margin-right</td>
<td>margin-top</td>
</tr>
<tr>
<td>outline</td>
<td>outline-color</td>
<td>outline-style (solid, dotted, dashed only)</td>
<td>outline-width</td>
</tr>
<tr>
<td>padding</td>
<td>padding-bottom</td>
<td>padding-left</td>
<td>padding-right</td>
</tr>
<tr>
<td>padding-top</td>
<td>page-break-after</td>
<td>page-break-before</td>
<td>page-break-inside</td>
</tr>
<tr>
<td>position</td>
<td>right</td>
<td>text-align (left, right, and center)</td>
<td>text-decoration</td>
</tr>
<tr>
<td>text-indent</td>
<td>top</td>
<td>unicode-bidi</td>
<td>vertical-align</td>
</tr>
<tr>
<td>visibility</td>
<td>white space (normal, nowrap only)</td>
<td>width</td>
<td>z-index</td>
</tr>
</tbody>
</table>

Using an image file URL
For optimal performance and reliability, Adobe recommends that you specify a local file URL for images stored on the server. In the following example, the `cfdocument` tag requests the server for images over HTTP even though the image files are stored locally:

```cftags
<cfdocument format="PDF">
  <table>
    <tr>
      <td>bird</td>
      <td><image src="images/bird.jpg"></td>
    </tr>
    <tr>
      <td>fruit</td>
      <td><image src="images/fruit.jpg"></td>
    </tr>
    <tr>
      <td>rose</td>
      <td><image src="images/rose.jpg"></td>
    </tr>
  </table>
</cfdocument>
```

Also, in some applications, the browser displays a Red X image error instead of the image in the browser. For better performance, and to avoid Red X image errors, set the `localUrl` attribute to `yes`:

```cftags
<cfdocument localUrl="yes" format="PDF">
  <table>
    <tr>
      <td>bird</td>
      <td><image src="images/bird.jpg"></td>
    </tr>
    <tr>
      <td>fruit</td>
      <td><image src="images/fruit.jpg"></td>
    </tr>
    <tr>
      <td>rose</td>
      <td><image src="images/rose.jpg"></td>
    </tr>
  </table>
</cfdocument>
```

**Scope variables**

When you use the `cfdocument` tag, ColdFusion creates a scope named `cfdocument`. This scope contains the following variables:

- `currentpagenumber`
- `totalpagecount`
- `totalsectionpagecount`
- `currentsectionpagenumber`

ColdFusion lets you use the scope variables inside any expression within a `cfdocumentitem` tag. For example, you can use the `currentpagenumber` variable to place the section name on even pages and the
chapter name on odd pages in the header, as follows:

```html
<cfdocument format="flashpaper">
    <cfdocumentitem type="header" evalAtPrint="true">
        <cfif (cfdocument.currentpagenumber mod 2) is 0>
            <cfoutput>#cfdocument.totalpagecount#</cfoutput>
        <cfelse>
            <cfoutput>#cfdocument.currentpagenumber#</cfoutput>
        </cfif>
    </cfdocumentitem>
    ...
</cfdocument>
```

If you define the `cfdocumentsection` tag within the `cfdocument` tag, then specify the `totalsectionpagecount` variable as follows:

```html
<cfdocument format="pdf">
    <cfdocumentitem type="header" evalatprint="true">
        <cfif (cfdocument.currentpagenumber mod 2) is 0>
            <cfoutput>#cfdocument.totalpagecount#</cfoutput>
        <cfelse>
            <cfoutput>#cfdocument.currentpagenumber#</cfoutput>
        </cfif>
    </cfdocumentitem>
    <cfdocumentitem>
        <cfoutput>cfdocument.currentpagenumber :#cfdocument.currentpagenumber#</cfoutput>
        <cfoutput>cfdocument.totalpagecount :#cfdocument.totalpagecount#</cfoutput>
        <cfoutput>cfdocument.totalsectionpagecount :#cfdocument.totalsectionpagecount#</cfoutput>
        <cfoutput>cfdocument.currentsectionpagenumber :#cfdocument.currentsectionpagenumber#</cfoutput>
    </cfdocumentitem>
    <cfdocumentitem type="footer" evalatprint="true">
        <cfif ! (cfdocument.currentpagenumber mod 2) is 0>
            <cfoutput>if#cfdocument.totalpagecount#</cfoutput>
        <cfelse>
            <cfoutput>else#cfdocument.currentpagenumber#</cfoutput>
        </cfif>
    </cfdocumentitem>
    <cfdocumentsection >Example Text
    </cfdocumentsection>
</cfdocument>
```

**Bookmarks**

ColdFusion 9 supports bookmarks. In the `cfdocument` tag, set the `bookmark` attribute to `yes`. Then specify the bookmark name for each `cfdocumentsection` tag.

The following example shows how to specify bookmarks for document sections:
Example

Example 1

<!--- This example creates a FlashPaper document. --->
<cfdocument format="flashpaper">
<p>This is a document rendered by the cfdocument tag.</p>
<table width="50%" border="2" cellspacing="2" cellpadding="2">
<tr>
<td><strong>Name</strong></td>
<td><strong>Role</strong></td>
</tr>
<tr>
<td>Bill</td>
<td>Lead</td>
</tr>
<tr>
<td>Susan</td>
<td>Principal Writer</td>
</tr>
<tr>
<td>Adelaide</td>
<td>Part Time Senior Writer</td>
</tr>
<tr>
<td>Thomas</td>
<td>Full Time for 6 months</td>
</tr>
<tr>
<td>Michael</td>
<td>Full Time for 4 months</td>
</tr>
</table>
</cfdocument>

Example 2
<!--- The following example shows how to use the cfdocument scope variables to generate section numbers and page numbers. --->

<cfdocument format="pdf">
<cfdocumentitem type="header" evalatprint="true">
<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr><td align="right"><cfoutput>#cfdocument.currentsectionpagenumber# of #cfdocument.totalsectionpagecount#</cfoutput></td></tr>
</table>
</cfdocumentitem>

<cfdocumentitem type="footer" evalatprint="true">
<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr><td align="center"><cfoutput>#cfdocument.currentpagenumber# of #cfdocument.totalpagecount#</cfoutput></td></tr>
</table>
</cfdocumentitem>

<cfdocumentsection>
<h1>Section 1</h1>
<cfloop from=1 to=50 index="i">
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.<p>
</cfloop>
</cfdocumentsection>

<cfdocumentsection>
<h1>Section 2</h1>
<cfloop from=1 to=50 index="i">
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.<p>
</cfloop>
</cfdocumentsection>

<cfdocumentsection>
<h1>Section 3</h1>
<cfloop from=1 to=50 index="i">
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.<p>
</cfloop>
</cfdocumentsection>
</cfdocument>
cfdocumentitem

Description

Specifies action items for a PDF or FlashPaper document created by the cfdocument tag. Action items include the following:

- header
- footer
- pagebreak

Category

Data output tags

Syntax

```xml
<cfdocument ...>
  <cfdocumentitem
      type = "pagebreak|header|footer"
      evalAtPrint = "true"
      header/footer text
    </cfdocumentitem>
</cfdocument>
```

Note

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also

cfreport, cfdocument, cfdocumentsection

History

ColdFusion 9: Added the evalAtPrint attribute.

ColdFusion 8: Added support for cfdocument.currentpagenumber, cfdocument.totalpagecount, cfdocument.totalsectionpagecount, and cfdocument.currentsectionpagenumber scopevariables.

ColdFusion MX 7.01: Added the src, srcfile, and mimetype attributes. ColdFusion MX 7: Added this tag.

Attributes

<p>| Attribute | Req/Opt | Default | Description |
|-----------|---------|---------|-------------|-------------|</p>
<table>
<thead>
<tr>
<th>type</th>
<th>Required</th>
<th>Specifies the action:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>- pagebreak: starts a new page at the location of the tag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- header: uses the text between the <code>&lt;cfdocumentitem&gt;</code> and <code>&lt;/cfdocumentitem&gt;</code> tags as the running header.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- footer: uses the text between the <code>&lt;cfdocumentitem&gt;</code> and <code>&lt;/cfdocumentitem&gt;</code> tags as the running footer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>evalAtPrint</th>
<th>Optional</th>
<th>false</th>
<th>A Boolean value that determines if the contents of the <code>cfdocumentitem</code> tag body has to be evaluated at the time of printing the document.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>- true: evaluates the contents of the <code>cfdocumentitem</code> tag body only at the time of printing the document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- false: evaluates the contents of the <code>cfdocumentitem</code> tag body immediately.</td>
</tr>
</tbody>
</table>

**Usage**

Use the `evalAtPrint` tag to evaluate the contents of the document before printing and to also accept additional attributes. Use the `cfdocumentitem` tag to control the formatting of a PDF or FlashPaper report. This tag must be wrapped inside a `<cfdocument>` `<cfdocument>` pair. Write code for one `cfdocumentitem` tag for each page break, running header, or running footer. ColdFusion has added support for `cfdocument` scope variables within the `cfdocumentitem` tag. You can use the `cfdocument` scope variable, `cfdocument.currentpagenumber`, to display the current page number in a header or footer. You can also use `{{cfdocument.totalpagecount}}` to display the total number of pages, for example:

```coldfusion
...  
  <cfdocumentitem type= "footer">
    #cfdocument.currentpagenumber# of #cfdocument.totalpagecount#
  </cfdocumentitem>
...```

For an example that uses the `cfdocument.totalsectionpagecount` and `cfdocument.currentsectionpagename` scope variables, see `cfdocument`.  

© 2014 Adobe Systems Incorporated. All rights reserved.
You can use `cfdocumentitem` tags with or without the `cfdocumentsection` tag, as follows:

- **Without `cfdocumentsection`** The `cfdocumentitem` attribute applies to the entire document, as follows:
  - If the tag is at the top of the document, it applies to the entire document.
  - If the tag is in the middle of the document, it applies to the rest of the document.
  - If the tag is at the end of the document, it has no affect.

- **With `cfdocumentsection` tags** The `cfdocumentitem` attribute applies only to the section and overrides previously specified header and footer specifications.

Example

```cfc
<cfquery datasource="cfdocexamples" name="parksQuery">
  SELECT parkname, suptmgr from parks
</cfquery>

<cfdocument format="PDF">
<cfdocumentitem type="header">National Parks Report</cfdocumentitem>
<!--- Use a footer with current page of totalpages format. --->
<cfdocumentitem type="footer">
<cfoutput>Page #cfdocument.currentpagenumber# of #cfdocument.totalpagecount#</cfoutput>
</cfdocumentitem>

<h1>Park list</h1>
<table width="95%" border="2" cellspacing="2" cellpadding="2">
<tr>
<th>Park</th>
<th>Manager</th>
</tr>
<cfoutput query="parksQuery">
<tr>
<td><font size="-1">#parkname#</font></td>
<td><font size="-1">#suptmgr#</font></td>
</tr>
</cfoutput>
</table>
</cfdocument>
**cfdocumentsection**

**Description**

Divides a PDF or FlashPaper document into sections. By using this tag in conjunction with a `cfdocumentitem` tag, each section can have unique headers, footers, and page numbers.

**Category**

*Data output tags*

**Syntax**

```cfc
<cfdocument ...
<cfdocumentsection
  authPassword = "authentication password"
  authUser = "authentication user name"
  marginBottom = "number"
  marginLeft = "number"
  marginRight = "number"
  marginTop = "number"
  mimeType = "text/plain|application/xml|image/jpeg|image/png|image/bmp|image/gif"
  name = "bookmark for the section"
  src = "URL|path relative to web root"
  srcfile = "absolute path of file"
  userAgent = "HTTP user agent identifier">
  HTML, CFML, and cfdocumentitem tags
</cfdocumentsection>
</cfdocument>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

`cfreport`, `cfdocument`, `cfdocumentitem`

**History**

ColdFusion 8: Added the `name`, `authPassword`, `authUser`, and `userAgent` attributes.
ColdFusion MX 7.01: Added the `src`, `srcfile`, and `mimetype` attributes.
ColdFusion MX 7: Added this tag and the `marginTop`, `marginBottom`, `marginLeft`, `marginRight` attributes.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<p>| <strong>authPassword</strong>  | Optional | Password sent to the target URL for Basic Authentication. Combined with <strong>username</strong> to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication. |
| <strong>authUser</strong>     | Optional | User name sent to the target URL for Basic Authentication. Combined with <strong>password</strong> to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerebos authentication. |
| <strong>marginBottom</strong> | Optional | Bottom margin in inches (default) or centimeters. To specify the bottom margin in centimeters, include the <code>unit=&quot;cm&quot;</code> attribute in the parent <code>cfdocument</code> tag. |
| <strong>marginLeft</strong>   | Optional | Left margin in inches (default) or centimeters. To specify the left margin in centimeters, include the <code>unit=&quot;cm&quot;</code> attribute in the parent <code>cfdocument</code> tag. |
| <strong>marginRight</strong>  | Optional | Right margin in inches (default) or centimeters. To specify the right margin in centimeters, include the <code>unit=&quot;cm&quot;</code> attribute in the parent <code>cfdocument</code> tag. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>marginTop</td>
<td>Optional</td>
<td>Top margin in inches (default) or centimeters. To specify the top margin in centimeters, include the <code>unit=&quot;cm&quot;</code> attribute in the parent <code>cfdocument</code> tag.</td>
</tr>
</tbody>
</table>
| mimeType      | Optional  | MIME type of the source document. Supported MIME types are:  
- `text/html`  
- `text/plain`  
- `application/xml`  
- `image/jpeg`  
- `image/png`  
- `image/gif`  
If you do not specify this attribute explicitly, ColdFusion uses the filename to determine the MIME type. |
| name          | Optional  | Bookmark name for the section.                                                                                                                                                                           |
| src           | Optional  | URL or the relative path to the web root. You cannot specify both the `src` and `srcfile` attributes.                                                                                                       |
| srcfile       | Optional  | Absolute path of an on-disk or in-memory file that is on the server. You cannot specify both the `src` and `srcfile` attributes.                                                                         |
| userAgent     | Optional  | Text to put in the HTTP User-Agent request header field. Used to identify the request client software.                                                                                                  |

**Usage**

Use the `cfdocumentsection` tag to divide a report into sections. Within each `cfdocumentsection` tag, you can use one or more `cfdocumentitem` tags to specify unique headers and footers for each section.

When using `cfdocumentsection`, ColdFusion ignores HTML and CFML not enclosed within `cfdocumentsection` tags.
The margin attributes override margins specified in previous sections or in the parent `cfdocument` tag. If you specify margin attributes, the units are controlled by the `unit` attribute of the parent `cfdocument` tag; the `unit` attribute has a default value of inches. The `cfdocumentsection` tag forces a page break so that each section starts on a new page. ColdFusion has added the `name` attribute to support bookmarks. Bookmarks defined at the `documentsection` tag level are children of the `cfdocument` root.

Example

Example 1

```cfc
<cfquery datasource="cfdocexamples" name="empSalary">
SELECT Emp_ID, firstname, lastname, e.dept_id, salary, d.dept_name
FROM employee e, departmt d
WHERE e.dept_id = d.dept_id
ORDER BY d.dept_name
</cfquery>

<cfdocument format="PDF">
<cfoutput query="empSalary" group="dept_id">
  <cfdocumentsection>
    <cfdocumentitem type="header">
      <font size="-3"><i>Salary Report</i></font>
    </cfdocumentitem>
    <cfdocumentitem type="footer">
      <font size="-3">Page #cfdocument.currentpagenumber#</font>
    </cfdocumentitem>
    <h2>#dept_name#</h2>
    <table width="95%" border="2" cellspacing="2" cellpadding="2">
      <tr>
        <th>Employee</th>
        <th>Salary</th>
      </tr>
      <cfset deptTotal = 0 >
      <!--- inner cfoutput --->
      <cfoutput>
      <tr>
        <td><font size="-1">#empSalary.lastname#, #empSalary.firstname#</font>
        <td align="right"><font size="-1">#DollarFormat(empSalary.salary)#</font>
      </tr>
      <cfset deptTotal = deptTotal + empSalary.salary>
      </cfoutput>
      <tr>
        <td align="right"><font size="-1">Total</font></td>
        <td align="right"><font size="-1">#DollarFormat(deptTotal)#</font></td>
      </tr>
      <cfset deptTotal = 0>
    </table>
  </cfdocumentsection>
</cfoutput>
</cfdocument>
```
Example 2: Bookmarks

<!--- This example uses the name attribute to define bookmarks in a PDF document at
the
section level. --->
<cfdocument format="pdf" bookmark="yes">
<cfdocumentsection name="section 1">
<!--- Insert some HTML content here. --->
</cfdocumentsection>
<cfdocumentsection name="section 2">
<!--- Insert some HTML content here. --->
</cfdocumentsection>
</cfdocument>
**cfdump**

**Description**

Use the `cfdump` tag to get the elements, variables, and values of most kinds of ColdFusion objects. Useful for debugging. You can display the contents of simple and complex variables, objects, components, user-defined functions, and other elements. The `cfdump` now shows component properties defined by `cfproperty` when you dump a CFC. A new key called **PROPERTIES** has been added in the component dump, which is expanded, by default. The text format of `cfdump` also provides this information.

**Category**

Debugging tags, Variable manipulation tags

**Syntax**

```
<cfdump
  var = "#variable#"
  output = "browser|console|file"
  format = "text|html"
  abort = "true|false">
  label = "text"
  metainfo = "yes|no"
  top = "number of rows|number of levels"
  show = "columns|keys"
  hide = "columns|keys"
  keys = "number of keys to display for structures"
  expand = "yes|no"
  showUDFs = "yes|no">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfcookie, cfparam, cfsavecontent, cfschedule, cfset, cftimer, cfwddx

**History**

- ColdFusion 9: Added the attribute `abort`.
- ColdFusion 8: Added the `show`, `format`, `hide`, `keys`, `metainfo`, `output`, and `showUDFs` attributes.
- ColdFusion MX 7: Added the `top` attribute.
- ColdFusion MX 6.1: Added the ability to dump COM objects; it displays the methods and Get and Put properties typeinfo information for the object.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| var       | Required |       | Variable to display. Enclose a variable name in number signs. These kinds of variables yield meaningful cfdump output:  
- array  
- CFC  
- COM object  
- file object  
- Java object  
- simple  
- query  
- structure  
- UDF  
- wddx  
- xml |
| expand    | Optional | yes   | • yes: in Internet Explorer and Mozilla, expands views.  
• no: contracts expanded views. |
| format    | Optional | text  | Use with the output attribute to specify whether to save the results of a cfdump to a file in text or HTML format. |
| hide      | Optional | all   | For a query, this is a column name or a comma-delimited list of column names. For a structure, this is a key or a comma-delimited list of keys.  
If you specify a structure element that doesn't exist, ColdFusion ignores it and does not generate an error. |
<p>| keys      | Optional | 9999  | For a structure, the number of keys to display. |
| label     | Optional |       | A string; header for the dump output. Ignored if the value of the var attribute is a simple types. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metainfo</td>
<td>Optional</td>
<td>yes for query{{no}} for persistence CFCs</td>
<td>For use with queries and persistence CFCs. Includes information about the query in the cfdump results, including whether the query was cached, the execution time, and the SQL. Specify <code>metainfo=&quot;no&quot;</code> to exclude this information from the query result. For persistence CFCs, if <code>metainfo=&quot;yes&quot;</code>, returns property attributes such as getters and setters.</td>
</tr>
<tr>
<td>output</td>
<td>Optional</td>
<td>browser</td>
<td>Where to send the results of cfdump. The following values are valid:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- browser</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- console</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- filename</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The filename must include the full pathname of the file. You can specify an absolute path, or a path that is relative to the ColdFusion temporary directory. You can use the <code>GetTempDirectory()</code> function to determine the ColdFusion temporary directory.</td>
</tr>
<tr>
<td>show</td>
<td>Optional</td>
<td>all</td>
<td>For a query, this is a column name or a comma-delimited list of column names. For a structure, this is a key or a comma-delimited list of keys.</td>
</tr>
<tr>
<td>showUDFs</td>
<td>Optional</td>
<td>yes</td>
<td>- yes: includes UDFs, with the methods collapsed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no: excludes UDFs.</td>
</tr>
</tbody>
</table>
Optional 9999  The number of rows to display. For a structure, this is the number of nested levels to display.

Optional false  If this attribute is set to "true", it stops processing the current page at the tag location.

Usage

The expand/contract display capability is useful when working with large structures, such as XML document objects, structures, and arrays.

To display a construct, use code such as the following, in which myDoc is a variable of type XmlDocument:

```cfml
<cfif IsXmlDoc(mydoc) is "yes">
  <cfdump var="#mydoc#">
</cfif>
```

The tag output is color-coded according to data type.

If a table cell is empty, this tag displays "empty string".

Example

```cfml
<!--- This example shows how to use this tag to display the CGI scope as a structure: --->

<cfdump var="#cgi#"/> <!--- This displays information about file objects. --->
<cfscrip>
myfile = FileOpen("c:\temp\test1.txt", "read");
</cfscrip>
myfile refers to:
<cfdump var="#myfile/filepath#"/>
```
cfelse

Description

Used as the last control block in a cfif tag block to handle any case not identified by the cfif tag or a cfelseif tag.

Category

Flow-control tags

Syntax

```html
<cfif expression>
  HTML and CFML tags
  <cfelseif expression>
    HTML and CFML tags
  <cfelse>
    HTML and CFML tags
  </cfelse>
</cfif>
```

See also

cific, cfelseif, cfabort, cfbreak, cfexecute, cfexit, cflocation, cfloop, cfswitch, cfthrow, cftry

Usage

If the values of the expressions in the containing cfif tag and all cfelseif tags are no, ColdFusion processes the code between this tag and the cfif end tag. This tag must be inside a cfif tag block. It does not require an end tag. For more information and an example, see cff.
cfelseif

Description

Used as a control block in a cfif tag block to handle any case not identified by the cfif tag or a cfelseif tag.

Category

Flow-control tags

Syntax

```html
<cfif expression>
    HTML and CFML tags  <cfelseif expression>
    HTML and CFML tags  <cfelse>
    HTML and CFML tags </cfif>
```

See also

cfif, cfelse, cfabort, cfbreak, cfexecute, cfexit, cflocation, cfloop, cfswitch, cfthrow, cftry

Usage

If the value of the expression in this tag is yes, and the values of the expressions in the containing cfif tag and preceding cfelseif tags are no, ColdFusion processes the code between this tag and a following cfelseif or cfelse tag, or the cfif end tag and then skips to the code following the cfif end tag. Otherwise, ColdFusion skips the code.

This tag must be inside a cfif tag block. It does not require an end tag. For more information and an example, see cfif.
cferror

Description

Displays a custom HTML page when an error occurs. This lets you maintain a consistent look and feel among an application's functional and error pages.

Category

Exception handling tags, Extensibility tags, Application framework tags

Syntax

```xml
<cferror
    template = "template path"
    type = "exception|validation|request"
    exception = "exception type"
    mailTo = "e-mail address">
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfthrow, cfthrow, cftry, Handling Errors in the Developing ColdFusion Applications.

History

ColdFusion MX: Deprecated the monitor option of the exception attribute. It might not work, and might cause an error, in later releases.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template</td>
<td>Required</td>
<td>Relative path to the custom error page. (A ColdFusion page was formerly called a template.)</td>
<td></td>
</tr>
</tbody>
</table>
| type       | Required | Type of error that the custom error page handles. The type also determines how ColdFusion handles the error page. For more information, see *Specifying a custom error page* in *Specifying custom error messages with the cferror tag* in the *Developing ColdFusion Applications*.

- **exception**: an exception of the type specified by the `exception` attribute.
- **validation**: errors recognized by server-side type validation.
- **request**: any encountered error. |
<table>
<thead>
<tr>
<th>exception</th>
<th>Optional</th>
<th>any</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type of exception that the tag handles:

- **application:** application exceptions.
- **database:** database exceptions.
- **template:** ColdFusion page exceptions.
- **security:** security exceptions.
- **object:** object exceptions.
- **missingInclude:** missing include file exceptions.
- **expression:** expression exceptions.
- **lock:** lock exceptions.
- **custom_type:** developer-defined exceptions, defined in the `cfthrow` tag.
- **any:** all exception types.

For more information on exception types, see [cftry](#).

<table>
<thead>
<tr>
<th>mailTo</th>
<th>Optional</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>An e-mail address. This attribute is available on the error page as the variable <code>error.mailto</code>. ColdFusion does not automatically send anything to this address.</td>
</tr>
</tbody>
</table>

**Usage**

Use this tag to provide custom error messages for pages in an application. This lets you maintain a consistent look and feel within the application, even when errors occur. You generally embed this tag in your Application CFC or Application.cfm file to specify error-handling responsibilities for an entire application. You must put it in one of these files if you specify `type="validation"`; ColdFusion ignores it on any other page. The `cftry` and `cfcatch` tags provide a more interactive way to handle ColdFusion errors within a ColdFusion page than the `cferror` tag, but the `cferror` tag is a good safeguard against general errors. To ensure that error pages display successfully, avoid using the `cfencode` utility to encode pages that include the `cferror` tag.

**Page types**

The following table describes the types of errors you can specify and code you can use on the pages that handle these error type:
<table>
<thead>
<tr>
<th>Page type</th>
<th>Description</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>Dynamically invoked by the CFML language processor when it detects an unhandled exception condition. Uses the full range of CFML tags. Error variables must be in <code>cfoutput</code> tags.</td>
<td>Can handle specific exception types or display general information for exceptions.</td>
</tr>
<tr>
<td>Request</td>
<td>Includes the error variables described in the Error variables section. Cannot include CFML tags, but you can display values of the error variables by enclosing them in number signs (#), as in <code>#error.MailTo#</code>.</td>
<td>Use as a backup error handler to other error handling methods, including exception type.</td>
</tr>
<tr>
<td>Validation</td>
<td>Handles data input validation errors that occur when submitting a form that uses hidden form-field validation or onSubmit validation. Cannot include CFML tags, but you can display values of the error variables by enclosing them in number signs (#), as in <code>#Error.InvalidFields#</code>. Specify the validation error handler in the Application.cfc or Application.cfm file.</td>
<td>Handles hidden form-field or onSubmit format validation errors only.</td>
</tr>
</tbody>
</table>

**Error variables**

The exception-handling page specified in the `cerror` tag template attribute contains one or more error variables. ColdFusion substitutes the value of the error variable when an error displays. The following table lists error variables:

<table>
<thead>
<tr>
<th>Page type</th>
<th>Error variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation only</td>
<td>error.validationHeader</td>
<td>Validation message header text.</td>
</tr>
<tr>
<td></td>
<td>error.invalidFields</td>
<td>Unordered list of validation errors.</td>
</tr>
<tr>
<td></td>
<td>error.validationFooter</td>
<td>Validation message footer text.</td>
</tr>
<tr>
<td>Request and Exception</td>
<td>error.diagnostics</td>
<td>Detailed error diagnostics from ColdFusion.</td>
</tr>
<tr>
<td></td>
<td>error.mailTo</td>
<td>E-mail address (same as value in <code>cerror.MailTo</code>).</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>error.dateTime</td>
<td>Date and time when error occurred.</td>
<td></td>
</tr>
<tr>
<td>error.browser</td>
<td>Browser that was running when error occurred.</td>
<td></td>
</tr>
<tr>
<td>error.remoteAddress</td>
<td>IP address of remote client.</td>
<td></td>
</tr>
<tr>
<td>error.HTTPReferer</td>
<td>Page from which client accessed link to page where error occurred.</td>
<td></td>
</tr>
<tr>
<td>error.template</td>
<td>Page executing when error occurred.</td>
<td></td>
</tr>
<tr>
<td>error.generatedContent</td>
<td>The content generated by the page up to the point where the error occurred.</td>
<td></td>
</tr>
<tr>
<td>error.queryString</td>
<td>URL query string of client's request.</td>
<td></td>
</tr>
<tr>
<td><strong>Exception only</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>error.message</td>
<td>Error message associated with the exception.</td>
<td></td>
</tr>
<tr>
<td>error.rootCause</td>
<td>The root cause of the exception. This structure contains the information that is returned by a <code>cfcatch</code> tag. For example, for a database exception, the SQL statement that caused the error is in the <code>error.RootCause.Sql</code> variable. For Java exceptions, this variable contains the Java servlet exception reported by the JVM as the cause of the &quot;root cause&quot; of the exception.</td>
<td></td>
</tr>
<tr>
<td>error.tagContext</td>
<td>Array of structures containing information for each tag in the tag stack. The tag stack consists of each tag that is currently open.</td>
<td></td>
</tr>
<tr>
<td>error.type</td>
<td>Exception type.</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

If `type = "exception"`, you can substitute the prefix `cferror` for `Error`; for example, `cferror.diagnostics, cferror.mailTo, or cferror.dateTime`.

**Example**
<h3>cferror Example</h3>

<!--- Example of cferror call within a page. 
    NOTE: If you use cferror type="VALIDATION" you MUST put it in 
    Application.cfc or Application.cfm --->
<cferror type = "REQUEST"
template = "request_err.cfm"
mailTo = "admin@mywebsite.com">
<!--- This query calls a non-existent datasource, triggering an error to be handled. --->
<cfquery name="testQuery" datasource="doesNotExist">
    select * from nothing
</cfquery>

<!--- Example of the page (request_err.cfm) to handle this error. --->
<html>
<head>
    <title>We're sorry -- An Error Occurred</title>
</head>
<body>
<h2>We're sorry -- An Error Occurred</h2>
<p>
If you continue to have this problem, please contact #error.mailTo#
with the following information:
</p>
<ul>
    <li><b>Your Location:</b> #error.remoteAddress#
    </li>
    <li><b>Your Browser:</b> #error.browser#
    </li>
    <li><b>Date and Time the Error Occurred:</b> #error.dateTime#
    </li>
    <li><b>Page You Came From:</b> #error.HTTPReferer#
    </li>
    <li><b>Message Content</b>:
        <p>#error.diagnostics#</p>
    </li>
</ul>
cfexchangecalendar

Description

Creates, deletes, modifies, gets, and responds to Microsoft Exchange calendar events, and gets calendar event attachments.

History

ColdFusion 11: Added three new optional attributes to the cfexchangecalendar tag for managing secondary calendars. These attributes are applicable for get and create actions only and valid for exchange server version starting from 2010. Secondary calendars can be specified in the exchangecalendar tag either using secondary calendar folder ID or using the calendar folder path

ColdFusion 10: getUserAvailability, getRooms, getRoomsList

- Added the attribute serverVersion.

ColdFusion 8: Added this tag.

Category

Communications tags

Syntax

```xml
create
cfexchangecalendar
required
action = "create"
  event = "#event information structure#"
optional
  connection = "connection ID"
  result = "variable for event UID">

delete
cfexchangecalendar
required
action = "delete"
  uid = "event UID,event UID, ..."
optional
  connection = "connection ID"
  message = "string"
  notify = "yes|no">

deleteAttachments
cfexchangecalendar
required
action = "deleteAttachments"
  uid = "event UID"
optional
  connection = "connection ID">

get
cfexchangecalendar
required
action = "get"
```
name = "query identifier"
  optional
  connection = "connection ID">

getAttachments
<cfexchangecalendar
required
action = "getAttachments"
name = "query identifier"
uid = "event UID"
optional
attachmentPath = "directory path"
connection = "connection ID">
generateUniqueFilenames = "no|yes"
getRooms
<cfexchangecalendar
action = "getRooms"
emailAddress = "e-mail_ address"
name = "name"
connection = "connection_ID" />
getRoomsList
<cfexchangecalendar
action = "getRoomList"
name = "name"
connection = "connection_ID" />
getUserAvailability
<cfexchangecalendar
action = "getUserAvailability"
attendees = "attendee_list"
connection = "connection_ID"
startDate = "date"
endDate = "date"
dataRequestType = "freeBusy|suggestions|freeBusyandSuggestions"
name = "name" />
modify
<cfexchangecalendar
required
action = "modify"
event = "#event information structure#"
uid = "event UID"
optional
connection = "connection ID">
respond
<cfexchangecalendar
required
action = "respond"
responseType = "accept|decline|tentative"
uid = "event UID"
optional
connection = "connection ID"
Note

For all actions, see `cfexchangeconnection` for additional attributes that you use if you do not specify the `connection` attribute. If you omit the `connection` attribute, create a temporary connection by specifying `cfexchangeconnection` tag attributes in the `cfexchangecalendar` tag. In this case, ColdFusion closes the connection when the tag completes. For details, see the `cfexchangeconnection` tag open action.

Note

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

See also

`cfexchangeconnection`, `cfexchangecontact`, `cfexchangefilter`, `cfexchangemail`, `cfexchangetask`, `Working with meetings and appointments` in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>N/A</td>
<td>Required</td>
<td></td>
<td>The action to take. Must be one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• create</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• deleteAttachments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• get</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getAttachment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getRooms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getRoomsList</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getUserAvailability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• modify</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• respond</td>
</tr>
<tr>
<td>attribute</td>
<td>function</td>
<td>type</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>attachmentPath</td>
<td>getAttachments</td>
<td>Optional</td>
<td>The filepath of the on-disk or in-memory directory in which to put the attachments. If an on-disk directory does not exist, ColdFusion creates it. Note: If you omit this attribute, ColdFusion does not save any attachments. If you specify a relative path, the path root is the ColdFusion temporary directory, which is returned by the GetTempDirectory function.</td>
<td></td>
</tr>
<tr>
<td>attendees</td>
<td>getUserAvailability</td>
<td>Required</td>
<td>Comma-separated list of all attendees.</td>
<td></td>
</tr>
<tr>
<td>connection</td>
<td>all</td>
<td>Optional</td>
<td>The name of the connection to the Exchange server, as specified in the cfexchangeconnection tag. If you omit this attribute, you must create a temporary connection by specifying cfexchangeconnection tag connection attributes in the cfexchangecalendar tag.</td>
<td></td>
</tr>
</tbody>
</table>
### availableAvailabilities

- **dataRequestType**
  - getUserAvailability
    - Required

- **getEmailAddress**
  - getRooms
    - Optional
    - Defines the Simple Mail Transfer Protocol (SMTP) address of a mailbox user.

- **getEvent**
  - createmodify
    - Required
    - A reference to the structure that contains the event properties to be set or changed, and their values. Specify this attribute in number signs (#). The event attribute also supports the categories key. For more information on the event structure, see Usage.

- **getEndDateTime**
  - getUserAvailability
    - Required
    - A string that ColdFusion can interpret as a date-time value.

- **getFreeBusy**
  - freeBusy
    - Returns an array of availability details.
  - Suggestions
    - Returns an array of struct that contains suggestion details.
  - freeBusyandSuggestions
    - Returns both the array of suggestions and an array of attendeeavailability.
    - See the sections on suggestion struct and attendeeavailability struct for details.
<table>
<thead>
<tr>
<th>getOccurrence</th>
<th>True/False</th>
<th>Optional</th>
<th>If True, retrieves all occurrences of a recurring event between specified startDateTime and endDate values and also, any single events. You cannot use cfExchangeFilter tag if you specify getOccurrence as True.</th>
</tr>
</thead>
<tbody>
<tr>
<td>generateUnique Filenames</td>
<td>getAttachments</td>
<td>Optional</td>
<td>no A Boolean value that specifies whether to generate unique filenames if multiple attachments have the same filenames. If two or more attachments have the same filename and this option is yes, ColdFusion appends a number to the filename body (before the extension) of any conflicting filenames. Thus, if three attachments have the name myfile.txt, ColdFusion saves the attachments as myfile.txt, myfile1.txt, and myfile2.txt.</td>
</tr>
<tr>
<td>message</td>
<td>deleterespond</td>
<td>Optional</td>
<td>The text of an optional message to send in the response or deletion notification.</td>
</tr>
<tr>
<td>name</td>
<td>getAttachmentsgetUserAvailabilitygetRoomsListgetRooms</td>
<td>Required</td>
<td>The name of the ColdFusion query variable that contains the retrieved events or information about the attachments that were retrieved. For more information on the returned data, see Usage.</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>notify</td>
<td>deleterespond</td>
<td>Optional</td>
<td>true</td>
</tr>
<tr>
<td>responseType</td>
<td>respond</td>
<td>Required</td>
<td>Must be one of the following values:</td>
</tr>
<tr>
<td>result</td>
<td>create</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>serverVersion</td>
<td>Optional</td>
<td>2007</td>
<td>Specifies the Microsoft Exchange Server version. The values are:</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2007</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If you do not specify the details, 2007 is taken by default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The value you specify overrides the value that you specify at</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the application level.</td>
</tr>
<tr>
<td>startDate</td>
<td>getUserAvailability</td>
<td>Required</td>
<td>A string that ColdFusion can interpret as a date-time value.</td>
</tr>
<tr>
<td>uid</td>
<td>delete</td>
<td>Required</td>
<td>Case-sensitive Exchange UID value or values that uniquely identify</td>
</tr>
<tr>
<td></td>
<td>getAttachments</td>
<td></td>
<td>the event or events on which to perform the action. For the delete</td>
</tr>
<tr>
<td></td>
<td>modify</td>
<td></td>
<td>action, this attribute can be a comma-delimited list of UID values.</td>
</tr>
<tr>
<td></td>
<td>respond</td>
<td></td>
<td>The deleteAttachment, getAttachment, modify, and respond actions allow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>only a single UID value.</td>
</tr>
<tr>
<td>folderid</td>
<td>Optional</td>
<td></td>
<td>Specify the calendar folder ID from which the calendar items need</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>to be managed. If the folder ID is not a valid calendar, an error will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reported.</td>
</tr>
</tbody>
</table>
### Usage

The `cfexchangecalendar` tag manages calendar events on the Exchange server. Use the `cfexchangecalendar` tag to do the following actions:

- Create an appointment or meeting event. You can create all-day events.
- Delete one or more events.
- Get one or more events that conform to an optional set of filter specifications, such as the subject, sender or recipient ID, time received, and so on.
- Get the attachments for a specific event.
- Modify an existing event.
- Respond to an event.

To use this tag, you must have a connection to an Exchange server. If you are using multiple tags that interact with the Exchange server, such as if you are creating several contact records, use the `cfexchangeconnection` tag to create a persistent connection. Then specify the connection identifier in each `cfexchangecalendar` tag, or in any other ColdFusion Exchange tag, if you are also accessing tasks, contacts, or mail. Doing this eliminates the overhead of creating and closing the connection for each tag. Alternatively, you can create a temporary connection that lasts only for the time that ColdFusion processes the single `cfexchangecalendar` tag. To do this, specify the connection attributes directly in the `cfexchangecalendar` tag. For details on the connection attributes, see the `cfexchangeconnection` tag.

#### Note

To create an Exchange calendar appointment, create a calendar event and do not specify any required or optional attendees.
The create action

When you specify the `create` action, the `event` attribute must specify a structure that contains the information that defines the events. The structure can have the following entries:

<table>
<thead>
<tr>
<th>Element</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllDayEvent</td>
<td>no</td>
<td>A Boolean value that indicates whether this is an all-day event.</td>
</tr>
<tr>
<td>Attachments</td>
<td></td>
<td>One or more paths to the files to send as attachments. Separate filepaths with semicolons (;) for Windows, and colons (:) for UNIX and Linux. Paths to the attachments must be absolute. If you specify one or more attachments for a <code>modify</code> action, the specified attachments are added to any existing attachments; the pre-existing attachments are not deleted.</td>
</tr>
<tr>
<td>Categories</td>
<td></td>
<td>A comma-delimited list of categories. The filter searches for events that match all the categories in the list.</td>
</tr>
<tr>
<td>Duration</td>
<td></td>
<td>The duration of the event in minutes.</td>
</tr>
<tr>
<td>EndTime</td>
<td></td>
<td>The end time of the event, in any valid ColdFusion date-time format.</td>
</tr>
<tr>
<td>Importance</td>
<td>normal</td>
<td>One of the following values: high, normal, low.</td>
</tr>
<tr>
<td>IsRecurring</td>
<td></td>
<td>A Boolean value that indicates whether this event repeats. If yes, specify a <code>RecurrenceType</code> element and elements to specify the recurrence details. For information on the recurrence fields, see the next table.</td>
</tr>
<tr>
<td>Location</td>
<td></td>
<td>A string that specifies the location of the event.</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Default</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Message</td>
<td>A string that contains a message about the event. The string can include HTML formatting.</td>
<td></td>
</tr>
<tr>
<td>OptionalAttendees</td>
<td>A comma-delimited list of mail IDs.</td>
<td></td>
</tr>
<tr>
<td>Organizer</td>
<td>A string that specifies the name of the meeting organizer.</td>
<td></td>
</tr>
<tr>
<td>Reminder</td>
<td>The time, in minutes before the event, at which to display a reminder message.</td>
<td></td>
</tr>
<tr>
<td>RequiredAttendees</td>
<td>A comma-delimited list of mail IDs.</td>
<td></td>
</tr>
<tr>
<td>Resources</td>
<td>A comma-delimited list of mail IDs for Exchange scheduling resources, such as conference rooms and display equipment.</td>
<td></td>
</tr>
<tr>
<td>Sensitivity</td>
<td>The valid values are normal, company-confidential, personal, and private.</td>
<td></td>
</tr>
<tr>
<td>StartTime</td>
<td>The start time of the event, in any valid ColdFusion date-time format. If you specify a date and time in this attribute and specify a YEARLY RecurrenceType with no other recurrence attributes, the event recurs yearly at the day and time specified in this attribute.</td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td>A string that describes the event subject.</td>
<td></td>
</tr>
</tbody>
</table>

The following table lists the elements that you use to specify the event recurrence if you set the IsRecurring field to a yes value. For a detailed description of how to specify event recurrence, see Specifying Calendar recurrence in Working with meetings and appointments in the Developing ColdFusion Applications.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecurrenceType</td>
<td>all</td>
<td>DAILY Used only if the structure has a yes IsRecurring element. Must be one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DAILY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• WEEKLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MONTHLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• YEARLY</td>
</tr>
<tr>
<td>RecurrenceNoEndDate</td>
<td>all</td>
<td>yes Boolean value; if yes, the event recurs until you change or delete the event. Cannot be used with RecurrenceCount or RecurrenceEndDate.</td>
</tr>
<tr>
<td>RecurrenceCount</td>
<td>all</td>
<td>The number of times the event recurs. Cannot be used with RecurrenceEndDate or RecurrenceNoEndDate.</td>
</tr>
<tr>
<td>RecurrenceEndDate</td>
<td>all</td>
<td>The date of the last recurrence. Cannot be used with RecurrenceCount or RecurrenceNoEndDate.</td>
</tr>
<tr>
<td>RecurrenceFrequency</td>
<td>DAILY, WEEKLY, MONTHLY</td>
<td>1 The frequency of the recurrence in days, weeks, or months, depending on the type. For example, for DAILY recurrence, a Recurrence Frequency of 3 schedules the event every three days.</td>
</tr>
<tr>
<td>RecurEveryWeekDay</td>
<td>DAILY</td>
<td>The recurrence of the event on every week day, but not on Saturday or Sunday. Cannot be used with RecurrenceFrequency.</td>
</tr>
</tbody>
</table>
RecurrenceDays | WEEKLY | The day or days of the week on which the event occurs. Must be one or more of the following values in a comma-delimited list: MON, TUE, WED, THU, FRI, SAT, {{SUN}}. If you omit this field for a weekly recurrence, the event recurs on the day of the week that corresponds to the specified start date.

RecurrenceDay | MONTHLY, YEARLY | The day of the week on which the event occurs. Must be one of the following values:
- MON
- TUE
- WED
- THU
- FRI
- SAT
- SUN

RecurrenceWeek | MONTHLY, YEARLY | The week of the month or year on which the event recurs. The valid values are:
- first
- second
- third
- fourth
- last

RecurrenceMonth | YEARLY | The month of the year on which the event recurs. The valid values are JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, and DEC.

The delete action

When you specify the delete action, specify a uid attribute with a comma-delimited list of one or more Exchange UIDs that identify the events to delete. Use the get action, with an appropriate filter expression, to determine the UID values to specify.

If all UIDs that you specify are invalid, the cfexchangecalendar tag generates an error. If at least one UID is valid, the tag ignores any invalid UIDs and deletes the items specified by the valid UID.
The get action

When you specify the get action, use child cfexchangefilter tags to specify the messages to get. For detailed information on filters, see cfexchangefilter. When the tag completes processing, the query object specified by the name attribute contains one record for each retrieved message. Each record has the following columns:

<table>
<thead>
<tr>
<th>AllDayEvent</th>
<th>Duration</th>
<th>EndTime</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasAttachment</td>
<td>HtmlMessage</td>
<td>Importance</td>
<td>IsRecurring</td>
</tr>
<tr>
<td>Location</td>
<td>Message</td>
<td>OptionalAttendees</td>
<td>Organizer</td>
</tr>
<tr>
<td>Reminder</td>
<td>RequiredAttendees</td>
<td>Resources</td>
<td>Sensitivity</td>
</tr>
<tr>
<td>StartTime</td>
<td>Subject</td>
<td>UID</td>
<td>Categories</td>
</tr>
</tbody>
</table>

The following table describes the From, HtmlMessage, Message, and UID fields. For detailed information on the other fields, see the table in the create action description.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>The Exchange ID of the person who created the event.</td>
</tr>
<tr>
<td>HtmlMessage</td>
<td>An HTML-formatted version of the message about the event.</td>
</tr>
<tr>
<td>Message</td>
<td>A plain-text version of the message about the event.</td>
</tr>
<tr>
<td>UID</td>
<td>The Exchange unique identifier for the mail event. Use this value to identify the event in the delete, getAttachments, and modify actions.</td>
</tr>
</tbody>
</table>

The getAttachments action

When you use the getAttachments action, specify a single UID and a name attribute. The cfexchangecalendar tag populates a query object with the specified name. Each record has the following information about an attachment to the event specified by the UID:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachmentFileName</td>
<td>The filename of the attachment.</td>
</tr>
<tr>
<td>attachmentFilePath</td>
<td>The absolute path of the attachment file on the server. If you omit the attachmentPath attribute, this column contains the empty string.</td>
</tr>
</tbody>
</table>
CID | The content-ID of the attachment. Typically used in HTML img tags to embed images in a message.
mimeType | The MIME type of the attachment, such as text/html.
isMessage | A Boolean value that specifies whether the attachment is a message.
size | The attachment size in bytes.

The tag places the attachments in the directory specified by the attachmentPath attribute. If you omit the attachmentPath attribute, ColdFusion does not get any attachments, it gets the information about the attachments. This lets you determine the event's attachments without incurring the overhead of getting the attachment files. Use the following syntax to specify an in-memory attachmentPath directory. In-memory files are not written to disk and speed processing of transient data.

```
attachmentpath = "ram:///path"
```

The path can include multiple directories, for example `ram:///petStore/orders/messageAttachments`. Create all directories in the path before you specify the file. For more information on using in-memory files, see *Working with in-memory files* in *Optimizing ColdFusion applications* in the *Developing ColdFusion Applications*. The getAttachments action works only if authentication for EWS (Exchange Web Services) is set to basic in the server setup of Exchange. IWA (Integrated Windows Authentication) is not supported.

**The modify action**

When you specify the modify action, you select the event to modify by specifying a uid attribute with single event UID; multiple UIDs are not allowed. You populate the event structure with only the fields that you are changing. For a detailed description of the fields and their valid values, see the table in the create action.

If an event has attachments and you specify attachments when you modify the event, the new attachments are added to the previous attachments; they do not replace them. Use the deleteAttachments action to remove any attachments.

**The respond action**

You use the respond action to respond to a meeting notification that you received by using the cfexchangemail tag. A meeting does not appear in your calendar, and cannot be accessed by using the cfexchangecalendar tag, until you respond to the mail message and accept or tentatively accept the request.

When you specify the respond action, specify the UID, from the notification mail message, of the event to which you are responding. Also specify the response type; that is, whether you are accepting, rejecting, or tentatively accepting the event. You can optionally specify a message to include in the response and set a flag whether to notify the creator of the event of your response. For detailed information on using the respond action, see *Working with meetings and appointments* in the *Developing ColdFusion Applications*.

**Suggestions struct values**

<table>
<thead>
<tr>
<th>Struct values</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th><strong>date</strong></th>
<th>Suggested day for the meeting.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>quality</strong></td>
<td>The quality of the suggested day, if Excellent, Good, Fair or Poor.</td>
</tr>
<tr>
<td><strong>TimeSuggestion</strong></td>
<td>An array of struct that contains the following values:</td>
</tr>
<tr>
<td></td>
<td>- <strong>MeetingDate</strong>: Suggested meeting time.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Quality</strong>: The quality of the time. It can be Excellent, Good, Fair, or Poor.</td>
</tr>
<tr>
<td></td>
<td>- <strong>array of conflicts</strong>: The conflicts at the suggested time. This is a struct that contains the following values: <strong>ConflictType</strong> (the type of conflict, which can be <strong>individualAttendeeConflict</strong>, which is a conflict with an attendee, <strong>GroupConflict</strong> which is a conflict with at least one member of a group, <strong>TooBigConflict</strong> which is a conflict with at least one member of a group, but the group was too big for detailed information to be returned, and <strong>UnknownAttendeeConflict</strong> which is a conflict with an unresolvable attendee or an attendee that is not a user, group, or contact).</td>
</tr>
<tr>
<td></td>
<td>- <strong>FreeBusyStatus</strong>: Gets the free/busy status of the conflicting attendee. Only meaningful when <strong>ConflictType</strong> is equal to <strong>IndividualAttendee</strong>. The values are Free, Tentative, Busy, OOF (time slot associated with the appointment appears as Out of Office) or NoData (no free/busy status is associated with the appointment).</td>
</tr>
<tr>
<td></td>
<td>- <strong>NoOfMembers</strong>: Gets the number of users, resources, and rooms in the conflicting group. Only meaningful when <strong>ConflictType</strong> is equal to <strong>GroupConflict</strong>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>NoOfMembersAvailable</strong>: Gets the number of available members (whose status is Free) in the conflicting group. Only meaningful when <strong>ConflictType</strong> is equal to <strong>GroupConflict</strong>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>NoOfMembersWithConflict</strong>: Gets the number of members who have a conflict (whose status is Busy, OOF, or Tentative) in the conflicting group. Only meaningful when <strong>ConflictType</strong> is equal to <strong>GroupConflict</strong>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>NoOfMembersWithNoData</strong>: Gets the number of members who do not have published free/busy data in the conflicting group. Only meaningful when <strong>ConflictType</strong> is equal to <strong>GroupConflict</strong>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>isWorkTime</strong>: If the suggested meeting happens in the work hours.</td>
</tr>
</tbody>
</table>
### Values of the struct attendeeavailability

<table>
<thead>
<tr>
<th>Struct values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalendarEvent</td>
<td>A struct that contains the following values:</td>
</tr>
<tr>
<td></td>
<td>• startTime: The start date and time of the event.</td>
</tr>
<tr>
<td></td>
<td>• endTime: The end date and time of the event.</td>
</tr>
<tr>
<td></td>
<td>• freeBusyStatus: The free/busy status associated with the event. It can have following values: Free, tentative, busy, OOF (Out of Office), or NoData (no free/busy status is associated with the appointment).</td>
</tr>
<tr>
<td></td>
<td>• details: The details of the calendar event; details is null if the user who requests for details does not have the appropriate rights. details is a struct that contains the following values: location (calendar location), eventstoreId (store ID of the calendar event), and Subject (can be isException which is a boolean value that indicates if the calendar event is an exception in a recurring series, isMeeting which is a boolean value that indicates if the calendar event is a meeting, isPrivate which is a boolean value that indicates if the calendar event is private, isRecurring which is a boolean value that indicates if the calendar event is recurring, and isReminderSet which is a boolean value that indicates if the calendar event has a reminder set).</td>
</tr>
<tr>
<td>mergedFreeBusyStatus</td>
<td>An array of struct that contains status. The status can be</td>
</tr>
<tr>
<td></td>
<td>• Free: The time slot associated with the appointment appears as Free.</td>
</tr>
<tr>
<td></td>
<td>• Tentative: The time slot associated with the appointment appears as Tentative.</td>
</tr>
<tr>
<td></td>
<td>• Busy: The time slot associated with the appointment appears as Busy</td>
</tr>
<tr>
<td></td>
<td>• OOF: The time slot associated with the appointment appears as Out of Office.</td>
</tr>
<tr>
<td></td>
<td>• NoData: No free/busy status is associated with the appointment.</td>
</tr>
<tr>
<td></td>
<td>• result: The result associated with the response. This can be success, warning, or error.</td>
</tr>
</tbody>
</table>
The free/busy view type retrieved for the attendee. It can have the following values:

- **None**: No view is returned. This value cannot be specified in a call to `GetUserAvailability`.
- **MergedOnly**: An aggregated free/busy stream. If the target user in one forest does not have an Availability service configured, the Availability service of the requestor retrieves the target user's free/busy information from the free/busy public folder. Because public folders only stores free/busy information in merged form, `MergedOnly` is the only available information.
- **FreeBusy**: The legacy status information (free, busy, tentative, and OOF). This also includes the start/end times of the appointments. This view is comprehensive than the legacy free/busy view because individual meeting start and end times are provided instead of an aggregated free/busy stream.
- **FreeBusyMerged**: All the properties in `FreeBusy` with a stream of merged free/busy availability information.
- **Detailed**: The legacy status information (free, busy, tentative, and OOF, the start/end times of the appointments; and various properties of the appointment such as subject, location, and importance). This requested view returns the maximum amount of information for which the requesting user is privileged. If merged free/busy information only is available, as with requesting information for users in a Microsoft Exchange Server 2003 forest, `MergedOnly` is returned. Otherwise, `FreeBusy` or `Detailed` are returned.
- **DetailedMerged**: Represents all the properties in `Detailed` with a stream of merged free/busy availability information. If, only merged free/busy information is available, for example if the mailbox exists on a computer running Exchange 2003, `MergedOnly` is returned. Otherwise, `FreeBusyMerged` or `DetailedMerged` is returned.
### workingHours

A struct that contains the following details:
- **startTime**: The start date and time of the event.
- **endTime**: The end date and time of the event.
- **daysOfTheWeek**: An array of struct. The working days of the attendees. It can have the following values: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Weekday, and Week EndDayDay.
- **Timezone**: A struct that contains the following fields: id (ID of the time zone definition) and name (name of this time zone definition).

### Exchange UID value

For all the cfexchangecalendar actions, the value of the attribute `uid` is as follows:

- If `exchangeServerVersion` is set to 2003 or 2007: The `uid` indicates the ID of the appointment in the mailbox of the organizer.
- If `exchangeServerVersion` is set to 2010: The `uid` indicates the ID of the received appointment in the mailbox of the attendee.

In the case of interaction with Microsoft Exchange server 2003 or 2007, whenever an appointment is created, the UID of the organizer can be used by the attendee for any operation such as responding, deleting, or getting attachments. In the case of Microsoft Exchange server 2010, the behavior is different. If attendees have to perform appointment-related actions, they have to first search for the appointment in their mailbox and then use the UID of that appointment.

### Example 1

The following example lets you create, and then modify a calendar event. When you first submit the form, ColdFusion creates the calendar event and redisplays the form with the data you entered. Accept the event before you modify the form and resubmit it. When you submit the form a second time, ColdFusion sends the modification information. For more information, see Working with meetings and appointments in the Developing ColdFusion Applications.

This example resends all the event data (to limit the example length), but you could change the example so that it only sends modified data.

```coldfusion
<!---- Create a structure to hold the event information. --->
<!---- A self-submitting form for the event information --->
<!---- This example omits recurrence to keep the code relatively simple --->
<cfparam name="form.eventID" default="0">

<!---- If the form was submitted, populate the event structure from it. --->
<cfif isDefined("Form.Submit")>
  <cfscript>
    sEvent.AllDayEvent="no";
    sEvent=StructNew();
    sEvent.Subject=Form.subject;
    if (IsDefined("Form.allDay")) {
      sEvent.AllDayEvent="yes";
      sEvent.StartTime=createDateTime(Year(Form.date), Month(Form.date),
        Day(Form.date), 8, 0, 0);
    } else {
```
sEvent.StartTime=createDateTime(Year(Form.date), Month(Form.date), Day(Form.date), Hour(Form.startTime), Minute(Form.startTime), 0);
sEvent.EndTime=createDateTime(Year(Form.date), Month(Form.date), Day(Form.date), Hour(Form.endTime), Minute(Form.endTime), 0);
}
sEvent.Location=Form.location;
sEvent.RequiredAttendees=Form.requiredAttendees;
sEvent.OptionalAttendees=Form.optionalAttendees;
//sEvent.Resources=Form.resources;
if (Form.reminder NEQ "") {
    sEvent.Reminder=Form.reminder;
} else {
    sEvent.Reminder=0;
}
sEvent.Importance=Form.importance;
sEvent.Sensitivity=Form.sensitivity;
sEvent.message=Form.Message;
</cfscript>

<!--- If this is the first time the form is being submitted
    Create a new event. --->
<cfif form.eventID EQ 0>
    <!--- Create the event in Exchange --->
    <cfexchangecalendar action="create"
        username="#user1#"  
        password="#password1#"
        server="#exchangeServerIP#"
        event="#sEvent#"
        result="theUID">
    </cfexchangecalendar>
    <!--- Output the UID of the new event. --->
    <cfif isDefined("theUID")>
        <cfoutput>Event Added. UID is#theUID#</cfoutput>
        <cfset Form.eventID = theUID >
    </cfoutput>
</cfif>
<cfelse>
    <!--- The form is being resubmitted with new data, so update the event. --->
    <cfexchangecalendar action="modify"
        username="#user1#"
        password="#password1#"
        server="#exchangeServerIP#"
        event="#sEvent#"
        uid="#Form.eventID#">
    </cfexchangecalendar>
    <cfoutput>Event ID #Form.eventID# Updated.</cfoutput>
</cfif>
</cfform>

<cfform format="xml" preservedata="yes" style="width:500" height="600">
    <cfinput type="text" label="Subject" name="subject" style="width:435"><br />
    <cfinput type="checkbox" label="All Day Event" name="allDay">
    <cfinput type="datefield" label="Date" name="date" validate="date" style="width:100">
    <cfinput type="text" label="Start Time" name="startTime" validate="time" style="width:100">
    <cfinput type="text" label="End Time" name="endTime" validate="time" style="width:100"><br />
    <cfinput type="text" label="Location" name="location" style="width:435"><br />
    <cfinput type="text" label="Required Attendees" name="requiredAttendees"
<cfinput type="text" label="Optional Attendees" name="optionalAttendees" style="width:435"><br />
<cfinput type="text" label="Resources" name="resources" style="width:435"><br />
<cfinput type="text" label="Reminder (minutes)" validate="integer" name="reminder" style="width:200">
<cfselect name="importance" label="Importance" style="width:100">
<option value="normal">Normal</option>
<option value="high">High</option>
<option value="low">Low</option>
</cfselect>
<cfselect name="sensitivity" label="Sensitivity" style="width:100">
<option value="normal">Normal</option>
<option value="company-confidential">Confidential</option>
<option value="personal">Personal</option>
<option value="private">Private</option>
</cfselect>
<cfinput type="textarea" label="Message" name="message" style="width:435; height:100">
<cfinput type="hidden" name="eventID" value="#Form.EventID#"/>
Example 2

The following example shows how you can get UserAvailability action:Application.cfm

```coldfusion
<!---- Setting the present Date ---->
<cfset todayDate = #Now()#>
<cfset fromDate = #DateFormat(todayDate, "mm/dd/yyyy")# & ' ' & #TimeFormat(todayDate, "HH:MM:SS")#>
<!---- Adding one day to present date----->
<cfset toDate1 = DateAdd("d", "1", "#fromDate#")>
<!----    <cfset toDate = #DateFormat(toDate, "mm/dd/yyyy")#&' '&#TimeFormat(toDate,
"HH:MM:SS")#>--->
<cfset toDate1 = #DateFormat(toDate1, "mm/dd/yyyy")# & ' ' & #TimeFormat(toDate1, "HH:MM:SS")#>
<!----Creating a calendar event ---->
<cffunction name="constructCalendarStruct" returntype="struct">
    <cfargument name="AllDayEvent" type="string"/>
    <cfargument name="Duration" type="string"/>
    <cfargument name="EndTime" type="string"/>
    <cfargument name="From" type="string"/>
    <cfargument name="HasAttachment" type="string"/>
    <cfargument name="HtmlMessage" type="string"/>
    <cfargument name="Importance" type="string"/>
    <cfargument name="IsRecurring" type="string"/>
    <cfargument name="Location" type="string"/>
    <cfargument name="Message" type="string"/>
    <cfargument name="OptionalAttendees" type="string"/>
    <cfargument name="Organizer" type="string"/>
    <cfargument name="Reminder" type="string"/>
    <cfargument name="RequiredAttendees" type="string"/>
    <cfargument name="Resources" type="string"/>
    <cfargument name="Sensitivity" type="string"/>
    <cfargument name="StartTime" type="string"/>
    <cfargument name="Subject" type="string"/>
    <cfargument name="UID" type="string"/>
    <cfscript>
        var eventInfo = structNew();
        if(isdefined("AllDayEvent") EQ 1)
            eventInfo.AllDayEvent = AllDayEvent;
        if(isdefined("Duration") EQ 1)
            eventInfo.Duration = Duration;
        if(isdefined("EndTime") EQ 1)
            eventInfo.EndTime = EndTime;
        if(isdefined("From") EQ 1)
            eventInfo.From = From;
        if(isdefined("HasAttachment") EQ 1)
            eventInfo.HasAttachment = HasAttachment;
        if(isdefined("HtmlMessage") EQ 1)
            eventInfo.HtmlMessage = HtmlMessage;
        if(isdefined("Importance") EQ 1)
            eventInfo.Importance = Importance;
        if(isdefined("IsRecurring") EQ 1)
```
eventInfo.IsRecurring = IsRecurring;
if(isdefined("Message") EQ 1)
   eventInfo.Message = Message;
if(isdefined("OptionalAttendees") EQ 1)
   eventInfo.OptionalAttendees = OptionalAttendees;
if(isdefined("Organizer") EQ 1)
   eventInfo.Organizer = Organizer;
if(isdefined("Reminder") EQ 1)
   eventInfo.Reminder = Reminder;
if(isdefined("RequiredAttendees") EQ 1)
   eventInfo.RequiredAttendees = RequiredAttendees;
if(isdefined("Resources") EQ 1)
   eventInfo.Resources = Resources;
if(isdefined("Sensitivity") EQ 1)
   eventInfo.Sensitivity = Sensitivity;
if(isdefined("StartTime") EQ 1)
   eventInfo.StartTime = StartTime;
if(isdefined("Subject") EQ 1)
   eventInfo.Subject = Subject;
if(isdefined("UID") EQ 1)
   eventInfo.UID = UID;
if(isdefined("Location") EQ 1)
   eventInfo.Location = Location;
</cfscript>
<cfreturn eventInfo>
</cffunction>
<!--- Function to delete calendar events --->
<cffunction name="deleteAllEvents">
<cfargument name="convariable" type="any"/>
<cfexchangecalendar action="get" name="deleteQuery" connection="convariable">
</cfexchangecalendar>
<cfloop query="deleteQuery">
   <cfexchangecalendar action="delete" connection=convariable uid=#deleteQuery.uid#>
</cfloop>
Availability.cfm

```
<cfexchangeConnection
    action="open"
    username = "user1"
    password="Password"
    server="IP_Address"
    serverversion="2010"
    connection="conn1">
<cfexchangeConnection
    action="open"
    username = "user2"
    password="Password"
    server="IP_Address"
    serverversion="2010"
    connection="conn2">
<cfset deleteAllEvents(conn1)>
<cfset deleteAllEvents(conn2)>
</cfexchangeConnection
<cfset deleteAllEvents(conn1)>
<cfset deleteAllEvents(conn2)>

<!---- Creating All day event for user1 --->
<cfset eventInfo = constructCalendarStruct(AllDayEvent="yes",Importance="High",Subject="Testplan1",StartTime="#fromDate#")>
<cfset eventInfo1 = constructCalendarStruct(AllDayEvent="yes",Importance="High",Subject="Testplan",StartTime="#fromDate#")>
<cfexchangeCalendar
    action="create"
    connection="conn1"
    event="#eventInfo#"
    result="UID"><cfscript>sleep(15000);</cfscript>
<cfexchangeCalendar
    action="create"
    connection="conn2"
    event="#eventInfo1#"
    result="UID"><cfscript>sleep(15000);</cfscript>
<cfexchangeCalendar
    attendees="a@cfadobe.com"
    action="getuseravailability"
    connection="conn1"
    datarequesttype="freebusyandsuggestions"
    startDate="#fromDate#"
    endDate="#toDate1#"
    name="result">
<cfdump var="#result#">
<cfset deleteAllEvents(conn1)>
<cfset deleteAllEvents(conn2)>
```

The following example shows how you can use the actions `getRooms` and `getRoomList`. 
Example 3

Creating and getting an appointment in a secondary calendar using the folderId attribute:

```coldfusion
<cfscript>
  eventStruct = structnew();
  eventStruct.AllDayEvent = "yes";
  eventStruct.Importance = "low";
  eventstruct.RequiredAttendees = "testing@abc.com";
  eventstruct.Subject = "Test Message";
  eventStruct.StartTime = now();
</cfscript>

<!--- Get Calendar Folder Id using exchangefolder tag --->
<cfexchangefolder action="GETEXTENDEDINFO" name="familyCalendar" connection="conn1" folderpath="Calendar/family">
  <!--- Must be a calendar folder--->
</cfexchangefolder>

<cfexchangecalendar action="create" event="#eventStruct#" connection="conn1" result="createdId" folderid="#familyCalendar.uid#">
  <cfoutput>
  Created Appointment Successfully in the given calendar
  </cfoutput>
</cfexchangecalendar>

<!--- getting appointments from the specified calendar using folderId--->
<cfexchangecalendar action="get" name="familyItems" connection="conn1" folderid="#familyCalendar.uid#">
  <cfexchangefilter name="subject" value="Test Message" />
</cfexchangecalendar>
<cfdump var="#familyItems#" />
```

Creating and getting an appointment in a secondary calendar using the folderpath attribute:
<cfexchangeconnection action="open" connection="conn1" server="<server hostname/IP>" username="<username>" password="<password>" serverversion="2010" protocol="<protocol>">

<cfscript>
    eventStruct = structnew();
    eventStruct.AllDayEvent = "yes";
    eventStruct.Importance = "low";
    eventStruct.RequiredAttendees = "testing@cfadobe.com";
    eventStruct.Subject = "This is from coldfusion just testing after the test " & rand("CFMX_COMPAT");
    eventStruct.StartTime = now();
</cfscript>

<!--- Default path delimiter is / no need to specify-->
<cfexchangecalendar action="create" event="#eventStruct#" connection="conn1" result="createdId" folderpath="Calendar/family">

<cfoutput>
    Created Appointment Successfully in the given calendar
</cfoutput>

<cfexchangecalendar action="get" name="familyItems" connection="conn1" folderpath="Calendar/family" pathDelimiter="/">
    <cfexchangefilter name="subject" value="This is from coldfusion just testing after the test" />
</cfexchangecalendar>

<cfdump var="#familyItems#">

</cfexchangeconnection>
**cfexchangeconnection**

**Description**

Opens or closes a persistent connection to a Microsoft Exchange server, or gets information about mailbox subfolders. You must have a persistent or temporary connection to use the `cfexchangecalendar`, `cfexchangecontact`, `cfexchangemail`, and `cfexchangetask` tags.

**History**

ColdFusion 10: Added the attribute serverVersion.

ColdFusion 8: Added this tag.

**Category**

[Communications tags](#)

**Syntax**
open
cfexchangeconnection
  required
  action = "open"
  connection = "connection ID">
  server = "Exchange server ID"
  username = "Exchange user ID">
  optional
    ExchangeApplicationName = "Application name"
    ExchangeServerLanguage = "Language name"
    formBasedAuthentication = "no|yes">
    formBasedAuthenticationURL = "URL">
    mailboxName = "Exchange mailbox">
    password = "user password"
    port = "IP port"
    protocol = "http|https"
    proxyHost = "proxy host URL"
    proxyPort = "proxy IP port"

getSubfolders
cfexchangeconnection
  required
  action = "getSubfolders"
  connection = "connection ID">
  name = "query name"
  optional
    folder = "Exchange folder path">
    recurse = "no|yes">
  OR
cfexchangeconnection
  required
  action = "getSubfolders"
  name = "query name"
  server = "Exchange server ID"
  username = "Exchange user ID">
  optional
    ExchangeApplicationName = "Application name"
    ExchangeServerLanguage = "Language name"
    folder = "Exchange folder path">
    formBasedAuthentication = "no|yes">
    formBasedAuthenticationURL = "URL">
    mailboxName = "Exchange mailbox">
    password = "user password"
    port = "IP port"
    protocol = "http|https"
    proxyHost = "proxy host URL"
    proxyPort = "proxy IP port"
    recurse = "no|yes">

close
cfexchangeconnection
  required
  action = "close"
  connection = "connection ID">
Note

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also

cfexchangecalendar, cfexchangecontact, cfexchangefilter, cfexchangemail, cfexchangetask; Managing connections to the Exchange server in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| action                   | all    | Required|         | The action to take. Must be one of the following values:  
|                          |        |         |         | • open: Open a new persistent named connection  
|                          |        |         |         | • close: Close a named connection  
<p>|                          |        |         |         | • getSubfolders: Get information about the subfolders of a specific folder. |
| connection               | all    | Required for open and close actions |         | The name of the connection. You can specify this ID in any tag that you use with the open connection. |
| ExchangeApplicationName  | opengetSubfolders | Optional | exchange | The name of the Exchange application to use in the URL that accesses the server. Specify this attribute if your IIS server does not use the default name for your Exchange application. |</p>
<table>
<thead>
<tr>
<th>ExchangeServerLanguage</th>
<th>opengetSubfolders</th>
<th>Optional</th>
<th>english</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The language of the Exchange server. If you are not sure, you can specify the empty string. For all values except english, including the empty string, the tag tries to get folder names from the server in the client's local language. In some cases, such as when there is a large amount of data on the server, it might take significant time to get folder names from Exchange server in the local language.

<table>
<thead>
<tr>
<th>folder</th>
<th>getSubfolders</th>
<th>Optional</th>
<th>The root of the mailbox</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The forward slash delimited path from the root of the mailbox to the folder for which to get subfolders. If a folder name contains a forward slash, use the `xF8F` escape sequence to specify the character in the name.
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formBasedAuthentication</td>
<td>Optional</td>
<td>A Boolean value that specifies whether to display a login form and use form based authentication when making the connection. If the attribute value is no (the default), and the Exchange server returns a 440 error status when ColdFusion tries to connect, ColdFusion displays the login form and attempts to use form based authentication. Therefore, you can safely omit this attribute if you do not know if the server requires form based authentication.</td>
</tr>
<tr>
<td>formBasedAuthenticationURL</td>
<td>Optional</td>
<td>The URL to which to post the user ID and password when an Exchange server uses form-based authentication. Use this attribute only if your Exchange server does not use default URL for form-based authentication. The default URL has the form <code>https://exchangeServer/exchweb/bin/auth/owaauth.dll</code>, for example, <code>{{https://exchange.mycompany.com/exchweb/bin/auth/owaauth.dll}}</code>.</td>
</tr>
<tr>
<td>mailboxName</td>
<td>opengetSubfolders</td>
<td>Optional</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------</td>
<td>----------</td>
</tr>
<tr>
<td>name</td>
<td>getSubfolders</td>
<td>Required</td>
</tr>
<tr>
<td>password</td>
<td>opengetSubfolders</td>
<td>Optional</td>
</tr>
<tr>
<td>port</td>
<td>opengetSubfolders</td>
<td>Optional</td>
</tr>
<tr>
<td>protocol</td>
<td>opengetSubfolders</td>
<td>Optional</td>
</tr>
<tr>
<td>proxyHost</td>
<td>opengetSubfolders</td>
<td>Optional</td>
</tr>
<tr>
<td>proxyPort</td>
<td>opengetSubfolders</td>
<td>Optional</td>
</tr>
</tbody>
</table>
| **recrise** | **getSubfolders** | **Optional** | **false** | A Boolean value:  
- **true**: get information on the immediate subfolders of the specified folder only.  
- **false**: get information on all levels of subfolders of the specified folder. |
| **server** | **opengetSubfolders** | **Required** |  | The IP address or URL of the server that is providing access to Exchange. |
| **serverVersion** |  | **Optional** | **2007** | Specifies the Microsoft Exchange Server version. The values are:  
- **2003**  
- **2007**  
- **2010**  
If you do not specify the details, 2007 is taken by default. The value you specify overrides the value that you specify at the application level. |
| **username** | **opengetSubfolders** | **Required** |  | The Exchange user ID. |

⚠️ **Note**

If you specify the **getSubfolders** action, you can specify the attributes that are listed as working for both the **open** and **getSubfolders** actions only if you do not specify a **connection** attribute.

**Usage**

The **cfexchangeconnection** tag can open or close a persistent connection with an Exchange server. If you use the **cfexchangeconnection** to open a connection before you use any **cfexchangecalendar**, **cfexchangecalendar**
tact, cfexchangemail, or cfexchangetask tags, you can use multiple tags to interact with the Exchange server without incurring the overhead of creating a connection for each tag.

⚠️ Note

To establish any connection, the Exchange server must grant you Outlook Web Access. For information on how to enable this access, see Enabling access to the Exchange server in Managing connections to the Exchange server in the Developing ColdFusion Applications. Also, you cannot establish a connection to an Exchange server if you require a special authentication step, such as requiring a VPN PIN or performing biometric authentication, on a server that is outside your firewall, and the authentication server then routes the messages to your Exchange server inside the firewall.

Use the cfexchangeconnection tag to close a persistent connection when you are finished accessing the Exchange server. If you do not close the connection, it remains open and does not time out. The cfexchangecalendar, cfexchangecontact, cfexchangemail, and cfexchangetask tags also let you specify the open action connection attributes (but not the connection attribute) to create a temporary connection that lasts for the duration of the single tag's activities, without requiring you to use the cfexchangeconnection tag to create the connection. In this case, ColdFusion automatically closes the connection when the tag completes processing.

The getSubfolders action can get information about the immediate subfolders of a specified folder (or of the top level of the mailbox), or information about all levels of subfolders. You must have a persistent connection to get the subfolders.

The query returned by the getSubfolders action has the following columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLDERNAME</td>
<td>The name of the subfolder, for example, ColdFusion.</td>
</tr>
<tr>
<td>FOLDERPATH</td>
<td>The forward slash delimited path to the folder from the mailbox root, including the folder name, for example, Inbox/Marketing/ColdFusion.</td>
</tr>
<tr>
<td>FOLDERSIZE</td>
<td>Size of the folder in bytes.</td>
</tr>
</tbody>
</table>

⚠️ Note

The ColdFusion exchange tags, including cfexchangeconnection use WebDAV to connect to the exchange server. HTTP access must be enabled on the exchange server to use the tags.

Example

The following example opens a connection, gets all mail sent from spamsource.com, and deletes the messages from the Exchange server:
<cfexchangeConnection
  action="open"
  username="#user1#"
  password="#password1#"
  server="#exchangeServerIP#"
  connection="testconn1">

<cfexchangemail action="get" name="spamMail" connection="testconn1">
  <cfexchangefilter name="fromID" value="spamsource.com">
</cfexchangemail>

<cfloop query="spamMail">
  <cfexchangeMail action="delete" connection="testconn1" uid="#spamMail.uid#">
</cfloop>

<cfexchangeConnection
  action="close"
  connection="testconn1">
cfexchangecontact

Description

Creates, deletes, modifies, and gets Microsoft Exchange contact records, and gets contact record attachments.

History

ColdFusion 10: Added the attribute serverVersion.

ColdFusion 8: Added this tag.

Category

Communications tags

Syntax
create
<cfexchangecontact
  required
  action = "create"
  contact = "#contact information structure#"
  optional
  connection = "connection ID"
  result = "variable for contact UID">

delete
<cfexchangecontact
  required
  action = "delete"
  uid = "contact UID,contact UID, ..."
  optional
  connection = "connection ID">

deleteAttachments
<cfexchangecontact
  required
  action = "deleteAttachments"
  uid = "contact UID"
  optional
  connection = "connection ID">

get
<cfexchangecontact
  required
  action = "get"
  name = "query identifier"
  optional
  connection = "connection ID">

getAttachments
<cfexchangecontact
  required
  action = "getAttachments"
  name = "query identifier"
  uid = "contact UID"
  optional
  attachmentPath = "directory path"
  connection = "connection ID"
  generateUniqueFilenames = "no\yes">

modify
<cfexchangecontact
  required
  action = "modify"
  contact = "#contact information structure#"
  uid = "contact UID"
  optional
  connection = "connection ID">
Note

If you omit the `connection` attribute, create a temporary connection by specifying `cfexchangeconnection` tag attributes in the `cfexchangecontact` tag. In this case, ColdFusion closes the connection when the tag completes. For details, see the `cfexchangeconnection` tag `open` action.

Note

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

See also

cfexchangecalendar, cfexchangeconnection, cfexchangefilter, cfexchangemail, cfexchangetask, Interacting with Microsoft Exchange Servers in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>N/A</td>
<td>Required</td>
<td></td>
<td>The action to take. Must be one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• create</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• deleteAttachments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• get</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getAttachments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• modify</td>
</tr>
<tr>
<td>attachmentPath</td>
<td>getAttachments</td>
<td>Optional</td>
<td></td>
<td>The absolute filepath of the directory in which to put the attachments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the directory does not exist, ColdFusion creates it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note</strong>: If you omit this attribute, ColdFusion does not save any</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>attachments.</td>
</tr>
<tr>
<td>attribute</td>
<td>value</td>
<td>type</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>connection</td>
<td>all</td>
<td>Optional</td>
<td>The name of the connection to the Exchange server, as specified in the <code>cfexchangeconnection</code> tag. If you omit this attribute, create a temporary connection by specifying <code>cfexchangeconnection tag connection open action attributes</code> in the <code>cfexchangecontact tag</code>.</td>
<td></td>
</tr>
<tr>
<td>contact</td>
<td>createmodify</td>
<td>Required</td>
<td>A reference to the structure that contains the <code>contact</code> properties to be set or changed and their values. Specify this attribute in number signs (#). For more information on the event structure, see Usage.</td>
<td></td>
</tr>
<tr>
<td>generateUniqueFilenames</td>
<td>getAttachments</td>
<td>Optional</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>----------</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A Boolean value that specifies whether to generate unique filenames if multiple attachments have the same filenames. If two or more attachments have the same filename and this option is yes, ColdFusion appends a number to the filename body (before the extension) of any conflicting filenames. Thus, if three attachments have the name myfile.txt, ColdFusion saves the attachments as myfile.txt, myfile1.txt, myfile2.txt, and myfile3.txt.

<table>
<thead>
<tr>
<th>name</th>
<th>getAttachments</th>
<th>Required</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The name of the ColdFusion query variable that contains the returned contact records or information about the attachments that were retrieved. For more information on the returned data, see Usage.

<table>
<thead>
<tr>
<th>result</th>
<th>create</th>
<th>Optional</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The name of a variable that contains the UID of the contact that is created. Use this value in the uid attribute of other actions to identify the contact to be acted on.
| serverVersion | Optional | 2007 | Specifies the Microsoft Exchange Server version. The values are:  
| | | | • 2003  
| | | | • 2007  
| | | | • 2010  
| | | | If you do not specify the details, 2007 is taken by default. The value you specify overrides the value that you specify at the application level.  
| uid | getAttachments, delete, modify | Required | A case-sensitive Exchange UID value that uniquely identifies the contacts on which to perform the action. For the delete action, this attribute can be a comma-delimited list of UID values. The deleteAttachment, getAttachment, and modify actions allow only a single UID value.  

When you specify the create or modify action, the contact attribute must specify a structure that contains information that defines the events. The structure can have the following elements. Include only the elements that you are setting or changing.  

<table>
<thead>
<tr>
<th>Assistant</th>
<th>Attachments</th>
<th>BusinessAddress</th>
<th>BusinessFax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessPhoneNumber</td>
<td>Categories</td>
<td>Company</td>
<td>Department</td>
</tr>
<tr>
<td>Description</td>
<td>DisplayAs</td>
<td>Email1</td>
<td>Email2</td>
</tr>
<tr>
<td>Email3</td>
<td>FirstName</td>
<td>HomeAddress</td>
<td>HomePhoneNumber</td>
</tr>
<tr>
<td>JobTitle</td>
<td>LastName</td>
<td>MailingAddressType</td>
<td>Manager</td>
</tr>
<tr>
<td>MiddleName</td>
<td>MobilePhoneNumber</td>
<td>NickName</td>
<td>Office</td>
</tr>
</tbody>
</table>
All fields except the BusinessAddress, HomeAddress, and OtherAddress fields contain text; the three address fields must contain structures with the following text fields:

- Street
- City
- State
- Zip
- Country

The Attachments field must contain the pathnames of any attachments to include in the contact. To specify multiple files, separate filepaths with semicolons (;) for Windows, and colons (:) for UNIX and Linux. Use absolute paths.

If you specify one or more attachments for a modify action, they are added to any existing attachments; the pre-existing attachments are not deleted.

The Categories field can have a comma-delimited list of the contact's categories.

If you do not specify a DisplayAs field, Exchange sets the display name to FirstName, LastName.

Usage

The `cfexchangecontact` tag manages contact records on the Exchange server. Use the `cfexchangecontact` tag to perform the following actions:

- Create a contact.
- Delete one or more contacts.
- Get one or more contact records that conform to an optional set of filter specifications, such as the last name, job title, or home phone number, and so on.
- Get the attachments for a specific contact record.
- Modify an existing contact.

To use this tag, you must have a connection to an Exchange server. If you are using multiple tags that interact with the Exchange server, such as if you are creating several contact records, use the `cfexchangeconnection` tag to create a persistent connection. You then specify the connection identifier in each `cfexchangecontact`, or any other ColdFusion Exchange tag, if you are also accessing tasks, contacts, or mail. Doing this eliminates the overhead of creating and closing the connection for each tag. Alternatively, you can create a temporary connection that lasts only for the time that ColdFusion processes the single `cfexchangecontact` tag. To do this, you specify the connection attributes directly in the `cfexchangecontact` tag. For details on the connection attributes, see the `cfexchangeconnection` tag open action.

attachmentPath attribute

Use the following syntax to specify an in-memory attachmentPath directory. In-memory files are not written to disk and speed processing of transient data.

```
attachmentpath = "ram:///filepath"
```

The path can include multiple directories, for example `ram:///petStore/orders/messageAttachments`. Create all directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications.
The delete action

When you specify the delete action you must specify a uid attribute with a comma-delimited list of one or more Exchange UIDs that identify the contacts to delete. You can use the get action, with an appropriate filter expression, to determine the UID values to specify.

If all UIDs that you specify are invalid, the cfexchangecontact tag generates an error. If at least one UID is valid, the tag ignores any invalid UIDs and deletes the items specified by the valid UID.

The get action

When you specify the get action, the query object specified by the name attribute contains one record for each retrieved contact. The query object has columns with the same names and data formats as the fields listed for the contact attribute structure, with the following changes:

- The query object has a Boolean HasAttachment column, and does not have an Attachments column. If the HasAttachment field is yes, use the getAttachments action to retrieve the attachments.
- The query object has an additional UID column with the unique identifier for the contact record in the Exchange server. Use this value in the uid attribute of the getAttachments, delete, and modify actions to identify the required record.
- The query object has an additional HtmlDescription column. The Description column has a plain-text version of the description, and the HtmlDescription column text includes the description's HTML formatting.

You use child cfexchangefilter tags to specify the messages to get. For detailed information, see cfexchangefilter.

The getAttachments action

When you use the getAttachments action, specify a single UID and a name attribute. The cfexchangecontact tag populates a query object with the specified name. Each record has the following information about an attachment to the contact specified by the UID:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachmentFileName</td>
<td>The filename of the attachment.</td>
</tr>
<tr>
<td>attachmentFilePath</td>
<td>The absolute path of the attachment file on the server. If you omit the attachmentPath attribute, this column contains the empty string.</td>
</tr>
<tr>
<td>CID</td>
<td>The content-ID of the attachment. Typically used in HTML img tags to embed images in a message.</td>
</tr>
<tr>
<td>mimeType</td>
<td>The MIME type of the attachment, such as text/html.</td>
</tr>
<tr>
<td>isMessage</td>
<td>A Boolean value specifying whether the attachment is a message.</td>
</tr>
<tr>
<td>size</td>
<td>The attachment size in bytes.</td>
</tr>
</tbody>
</table>

The tag places the attachments in the directory specified by the attachmentPath attribute. If you omit the attachmentPath attribute, ColdFusion does not get any attachments, it gets the information about the attachments. This lets you determine the attachments without incurring the overhead of getting the attachment files. Use the following
syntax to specify an in-memory attachmentPath directory. In-memory files are not written to disk and speed processing of transient data.

```cfc
attachmentpath = "ram:///path"
```

The path can include multiple directories, for example `ram:///petStore/orders/messageAttachments`. Create all directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications. The `getAttachments` action works only if authentication for EWS (Exchange Web Services) is set to basic in the server setup of Exchange. IWA (Integrated Windows Authentication) is not supported.

The modify action

If you specify the modify action, the uid attribute must specify a single Exchange UID. The contact structure must specify only the fields that you are changing. Any fields that you do not specify remain unchanged. If a contact has attachments and you specify attachments when you modify the contact, the new attachments are added to the previous attachments, and do not replace them. Use the `deleteAttachments` action to remove any attachments.

Example

The following example lets a user enter information in a form and creates a contact on the Exchange server with the information:

```cfc
<!--- Create a structure to hold the contact information. --->
<cfset sContact="#StructNew()#">

<!--- A self-submitting form for the contact information --->
<cfform format="flash" width="550" height="460">
  <cfformitem type="html"><b>Name</b></cfformitem>
  <cfformgroup type="horizontal" label="">
    <cfinput type="text" label="First" name="firstName" width="200">
    <cfinput type="text" label="Last" name="lastName" width="200">
  </cfformgroup>
  <cfformgroup type="VBox">
    <cfformitem type="html"><b>Address</b></cfformitem>
    <cfinput type="text" label="Company" name="Company" width="435">
    <cfinput type="text" label="Street" name="street" width="435">
    <cfinput type="text" label="City" name="city" width="200">
    <cfselect name="state" label="State" width="100">
      <option value="CA">CA</option>
      <option value="MA">MA</option>
      <option value="WA">WA</option>
    </cfselect>
    <cfinput type="text" label="Country" name="Country" width="200" Value="U.S.A.">
  </cfformgroup>
  <cfformitem type="html"><b>Phone</b></cfformitem>
  <cfformitem type="html" validate="telephone" label="Business" name="businessPhone" width="200">
  </cfformitem>
  <cfformitem type="html" validate="telephone" label="Mobile" name="cellPhone" width="200">
  </cfformitem>
  <cfformitem type="html" validate="telephone" label="Fax" name="fax" width="200">
  </cfformitem>
  <cfformitem type="html"><b>Email</b></cfformitem>
  <cfinput type="text" validate="email" name="email" width="200">
```
<cfformgroup>

<cfinput type="Submit" name="submit" value="Submit" />
</cfform>

<cfif isDefined("Form.Submit")>
  <cfscript>
    sContact.FirstName=Form.firstName;
    sContact.Company=Form.company;
    sContact.LastName=Form.lastName;
    sContact.BusinessAddress.Street=Form.street;
    sContact.BusinessAddress.City=Form.city;
    sContact.BusinessAddress.State=Form.state;
    sContact.BusinessAddress.Country=Form.country;
    sContact.BusinessPhoneNumber=Form.businessPhone;
    sContact.MobilePhoneNumber=Form.cellPhone;
    sContact.BusinessFax=Form.fax;
    sContact.Email1=Form.email;
  </cfscript>

  <!--- Create the contact in Exchange --->
  <cfexchangecontact action="create"
    username="#user1#"
    password="#password1#"
    server="#exchangeServerIP#"
    contact="#sContact#"
    result="#theUID#">
  </cfexchangecontact>

  <!--- Display a confirmation that the contact was added. --->
  <cfif isDefined("theUID")>
    <cfoutput>Contact Added. UID is#theUID#</cfoutput>
  </cfif>
</cfif>
</cfif>
</cfif>
**cfexchangeconversation**

**Description**

Helps users organize and manage conversations from a Microsoft Exchange account. The following actions are supported:

- Finds the required conversations in folder/subfolders based on filters.
- Status of the conversation; if read
- Copy, move, or delete conversation

**History**

ColdFusion 10: Added this tag.

**Category**

Communications tags

**Syntax**
get
<cfexchangeconversation
action = "get"
connection = "connection_ID"
folderID = "Exchange folder UID"
name = "query name"
</cfexchangeconversation>
setReadState
<cfexchangeconversation
action = "setReadState"
connection = "connection_ID"
folderID = "Folder_UID"
UID = "conversation_UID"
isRead = true|false
</cfexchangeconversation>
copy
<cfexchangeconversation
action = "copy"
connection = "connection_ID"
FolderID = "conversation_folder_UID"
UID = "conversation_UID"
destinationFolderID = "destination_folder_UID"
</cfexchangeconversation>
move
<cfexchangeconversation
action = "move"
connection = "connection_ID"
folderID = "conversation_folder_UID"
UID = "conversation_UID"
destinationFolderID = "destination_folder_UID"
</cfexchangeconversation>
delete
<cfexchangeconversation
action = "delete"
connection = "connection_ID"
folderID = "conversation_folder_UID"
UID = "conversation_UID"
deleteType = hardelete|softdelete|movetodeleteditems
</cfexchangeconversation>

See also
cfexchangecalendar, cfexchangeconnection, cfexchangefilter, cfexchangemail, cfexchangetask
Interacting with Microsoft Exchange Servers in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| action          | N/A     | Required | The action to take. Must be one of the following values:  
|                |         |          |  
|                |         |          | - get  
|                |         |          | - setReadState  
|                |         |          | - copy  
|                |         |          | - move  
|                |         |          | - delete  
| connection     | All actions | Required | The name of the connection to the Exchange server, as specified in the `{{cfexchangeconnection}}` tag.  
|                |         |          | If you omit this attribute, create a temporary connection by specifying `cfexchangeconnection` tag connection attributes in the `{{cfexchangecontact}}` tag.  
| name           | get     | Required | The name of the ColdFusion query variable that contains the returned conversation information.  
| folderID       | All actions | Required | A case-sensitive Exchange UID value that uniquely identifies the folder.  
| UID            | All     | Required | If `yes`, marks the conversation as read.  
<p>| isRead         | setReadState | Required | Indicates the status of the conversation, if read or not. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destinationFolder</td>
<td>move copy Required A case-sensitive Exchange UID value that uniquely identifies the destination folder.</td>
</tr>
<tr>
<td>deleteType</td>
<td>delete Required moveToDeletedItems • hardDelete: Removes a folder permanently from the store. • softDelete: Removes a folder to the dumpster, if dumpster is enabled. • moveToDeletedItems: Moves a folder to the deleted items folder.</td>
</tr>
</tbody>
</table>

Filter parameters for cfcollection action="get"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxRows</td>
<td>Specify the maximum number of conversations that have to be returned. Default is 100, and if -1 is specified, all conversations are returned.</td>
</tr>
<tr>
<td>catagories</td>
<td>A comma-separated list of categories stamped on messages in the conversation (only in the current folder).</td>
</tr>
<tr>
<td>flagStaus</td>
<td>The flag status for the conversation, calculated by aggregating individual message flag status in the current folder. It can have the following values: • NotFlagged • Flagged • Complete</td>
</tr>
<tr>
<td>GlobalCategories</td>
<td>A comma-separated list that summarizes the categories stamped on messages in the conversation, across all folders in the mailbox.</td>
</tr>
<tr>
<td>GlobalFlagStatus</td>
<td>The flag status for the conversation, calculated by aggregating individual message flag status across all folders in the mailbox. It can have the following values:</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
|                       | • NotFlagged  
|                       | • Flagged  
|                       | • Complete |
| GlobalHasAttachments   | A value that indicates if at least one message in the conversation, across all folders in the mailbox, has an attachment. |
| GlobalImportance      | The importance of this conversation, calculated by aggregating individual message's importance across all folders in the mailbox. It can have following values: |
|                       | • Low  
|                       | • Normal  
<p>|                       | • High |
| GlobalItemClasses     | A comma-separated list that summarizes the classes of the items in the conversation, across all folders in the mailbox. |
| GlobalItemIds         | A comma-separated list of IDs of the messages in the conversation, across all folders in the mailbox. |
| GlobalLastDeliveryTime | The delivery time of the message that was last received in the conversation across all folders in the mailbox. |
| GlobalMessageCount    | The total number of messages in the conversation across all folders in the mailbox. |
| GlobalSize            | The size of the conversation, calculated by adding the sizes of all messages in the conversation across all folders in the mailbox. |
| GlobalUniqueRecipients | A comma-separated list of recipients in the conversation across all folders in the mailbox. |
| GlobalUniqueSenders   | A comma-separated list of senders in the conversation across all folders in the mailbox. |
| GlobalUniqueUnreadSenders | A comma-separated list of senders whose messages are currently unread in the conversation across all folders in the mailbox. |
| GlobalUnreadCount     | The total number of unread messages in the conversation across all folders in the mailbox. |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasAttachments</td>
<td>Boolean value that indicates if at least one message in the conversation, in the current folder only, has an attachment.</td>
</tr>
<tr>
<td>U{{Id}}</td>
<td>UID of the conversation.</td>
</tr>
<tr>
<td>Importance</td>
<td>The importance of the conversation, calculated by aggregating individual message's importance (only in the current folder). It can have following values:</td>
</tr>
<tr>
<td></td>
<td>• Low</td>
</tr>
<tr>
<td></td>
<td>• Normal</td>
</tr>
<tr>
<td></td>
<td>• High</td>
</tr>
<tr>
<td>ItemClasses</td>
<td>A comma-separated list of classes of the items in the conversation (only in the current folder).</td>
</tr>
<tr>
<td>ItemIds</td>
<td>A comma-separated list of IDs of the messages in the conversation (only in the current folder). It is an array of ItemId objects.</td>
</tr>
<tr>
<td>LastDeliveryTime</td>
<td>The delivery time of the message that was last received in the conversation (in the current folder only).</td>
</tr>
<tr>
<td>MessageCount</td>
<td>The total number of messages in the conversation (in the current folder only).</td>
</tr>
<tr>
<td>size</td>
<td>The size of the conversation, calculated by adding the sizes of all messages in the conversation (in the current folder only).</td>
</tr>
<tr>
<td>topic</td>
<td>The topic of the conversation.</td>
</tr>
<tr>
<td>uniqueRecipients</td>
<td>A comma-separated list of message recipients in the conversation (in the current folder only)</td>
</tr>
<tr>
<td>uniqueSenders</td>
<td>A comma-separated list of senders in the conversation (in the current folder only).</td>
</tr>
<tr>
<td>uniqueUnreadSenders</td>
<td>A comma-separated list of senders whose messages are currently unread in the conversation (in the current folder only).</td>
</tr>
<tr>
<td>unreadCount</td>
<td>Total number of unread messages in the conversation (in the current folder only).</td>
</tr>
</tbody>
</table>

**Example**

The following example shows how you can perform the actions `get`, `setReadState`, `copy`, `move`, and `delete` on
conversations:

```coldfusion
<cfexchangeconnection action="open" username="conv" password="Password"
server="IP_Address" serverversion="2010" connection="conn1">
<!--- Finding information about Inbox --->
<cfexchangefolder action="getextendedinfo" connection="conn1" name="result"
folderpath="Inbox">
<cfexchangefolder action="getextendedinfo" connection="conn1" name="result1"
folderpath="Drafts">
<cfexchangeconversation action="get" folderid="#result.uid#" name="conversations"
connection="conn1">
<cfexchangeconversation action="copy" UID="#myArray[1]#" folderid="#result1.uid#"
destinationfolderid="#result1.uid#" connection="conn1">
<!--- Getting the detail about the conversation --->
<cfexchangeconversation action="get" folderid="#result1.uid#" name="conversations1"
connection="conn1">
<cfset myArray = ArrayNew(1)>
<cfloop query="conversations">
<cfset temp = ArrayAppend(myArray,="#UID#")>
</cfloop>
<!--- Copy the conversation to Drafts --->
<cfexchangeconversation action="copy" UID="#myArray[1]#" folderid="#result.uid#"
destinationfolderid="#result1.uid#" connection="conn1">
<cfset myArray = ArrayNew(1)>
<cfloop query="conversations">
<cfset temp = ArrayAppend(myArray,="#UID#")>
</cfloop>
<!--- Getting the detail about the conversation --->
<cfexchangeconversation action="get" folderid="#result1.uid#" name="conversations1"
connection="conn1">
<cfset myArray = ArrayNew(1)>
<cfloop query="conversations">
<cfset temp = ArrayAppend(myArray,="#UID#")>
</cfloop>
<!--- Moving the conversations--->
<cfexchangeconversation action="move" UID="#myArray[1]#" folderid="#result.uid#"
destinationfolderid="#result1.uid#" connection="conn1">
<!--- Getting the detail about the conversation --->
<cfexchangeconversation action="get" folderid="#result1.uid#" name="conversations2"
connection="conn1">
<cfset myArray = ArrayNew(1)>
<cfloop query="conversations">
<cfset temp = ArrayAppend(myArray,="#UID#")>
</cfloop>
<!--- Moving the conversation back to the initial location--->
<cfexchangeconversation action="move" UID="#conversations2.uid#" folderid="#result1.uid#"
destinationfolderid="#result.uid#" connection="conn1">
</cfexchangeconversation>
</cfexchangeconversation>
</cfdump>
```
**cfexchangefilter**

**Description**

Specifies filter parameters that control the actions of cfexchangemail, cfexchangecalendar, cfexchangetask, and cfexchangecontact, get operations.

**History**

ColdFusion 8: Added this tag.

**Category**

Communications tags

**Syntax**

```
<cfexchangefilter
   name = "filter type"
   value = "filter value">

OR

<cfexchangefilter
   name = "filter type"
   from = "date/time"
   to = "date/time">
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfexchangecalendar, cfexchangeconnection, cfexchangecontact, cfexchangemail, cfexchangetask, Getting Exchange items and attachments in the Developing ColdFusion Applications

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>The type of filter to use.</td>
</tr>
<tr>
<td><strong>from</strong></td>
<td>Optional</td>
<td>The start date or date/time combination of the range to use for filtering. Cannot be used with the <code>value</code> attribute. If you specify a <code>from</code> attribute without a <code>to</code> attribute, the filter selects for all entries on or after the specified date or time. The value can be in any date/time format recognized by ColdFusion, but must correspond to a value that is appropriate for the filter type.</td>
<td></td>
</tr>
<tr>
<td><strong>to</strong></td>
<td>Optional</td>
<td>The end date or date/time combination for the range used for filtering. Cannot be used with the <code>value</code> attribute. If you specify a <code>to</code> attribute without a <code>from</code> attribute, the filter selects for all entries on or before the specified date or time. The value can be in any date/time format recognized by ColdFusion, but must correspond to a value that is appropriate for the filter type.</td>
<td></td>
</tr>
</tbody>
</table>
The `cfexchangeCalendar` tag filters can have the following `name` attributes and associated `value`, `or` and `from` attributes that you use to specify the filter parameters for the specified action:

<table>
<thead>
<tr>
<th>name attribute</th>
<th>Specification attributes</th>
<th>Valid specification attribute values</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxRows</td>
<td>value</td>
<td>A positive integer specifying the maximum number of matching rows to return. By default, the maximum number of rows is 100. You can also specify -1; to return all matching rows.</td>
</tr>
<tr>
<td>allDayEvent</td>
<td>value</td>
<td>A Boolean value.</td>
</tr>
<tr>
<td>duration</td>
<td>value</td>
<td>An integer number of minutes.</td>
</tr>
<tr>
<td>endTime</td>
<td>from to</td>
<td>A string that ColdFusion can interpret as a date-time value.</td>
</tr>
<tr>
<td>fromID</td>
<td>value</td>
<td>An Exchange user ID.</td>
</tr>
<tr>
<td>hasAttachment</td>
<td>value</td>
<td>A Boolean value.</td>
</tr>
<tr>
<td>importance</td>
<td>value</td>
<td>One of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• high</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• low</td>
</tr>
<tr>
<td>isRecurring</td>
<td>value</td>
<td>A Boolean value.</td>
</tr>
</tbody>
</table>
### location

A string.

### message

A string.

### optionalAttendees

A comma-delimited list of Exchange user IDs.

### organizer

A string that identifies the organizer. This value does not need to be an Exchange ID or e-mail address.

### requiredAttendees

A comma-delimited list of Exchange user IDs.

### sensitivity

One of the following values:
- normal
- personal
- private
- confidential

### startTime

A string that ColdFusion can interpret as a date-time value.

### subject

A string.

### UID

A case-sensitive Exchange message UID that uniquely identifies one calendar entry.

The `cfexchangecontact` tag filters can have the following name attributes and associated value attributes. Unlike other tags, you do not use from or to attributes.

<table>
<thead>
<tr>
<th>name attribute</th>
<th>value attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxRows</td>
<td>A positive integer that specifies the maximum number of matching rows to return. By default, the maximum number of rows is 100. You can also specify -1; to return all matching rows.</td>
</tr>
<tr>
<td>assistant</td>
<td>A string.</td>
</tr>
<tr>
<td>businessAddress</td>
<td>A structure with the following fields: Street, City, State, Zip, Country.</td>
</tr>
<tr>
<td>businessFax</td>
<td>A string.</td>
</tr>
<tr>
<td>businessPhoneNumber</td>
<td>A string.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>categories</td>
<td>A comma-delimited list of categories. The filter searches for contacts that match all the categories in the list.</td>
</tr>
<tr>
<td>company</td>
<td>A string.</td>
</tr>
<tr>
<td>description</td>
<td>A string.</td>
</tr>
<tr>
<td>displayAs</td>
<td>A string.</td>
</tr>
<tr>
<td>email1</td>
<td>A string.</td>
</tr>
<tr>
<td>email2</td>
<td>A string.</td>
</tr>
<tr>
<td>email3</td>
<td>A string.</td>
</tr>
<tr>
<td>firstName</td>
<td>A string.</td>
</tr>
<tr>
<td>hasAttachment</td>
<td>A Boolean value.</td>
</tr>
<tr>
<td>homeAddress</td>
<td>A structure with the following fields: Street, City, State, Zip, Country.</td>
</tr>
<tr>
<td>homePhoneNumber</td>
<td>A string.</td>
</tr>
<tr>
<td>jobTitle</td>
<td>A string.</td>
</tr>
<tr>
<td>lastName</td>
<td>A string.</td>
</tr>
<tr>
<td>mailingAddressType</td>
<td>One of the following values: Home, Business, Other.</td>
</tr>
<tr>
<td>manager</td>
<td>A string.</td>
</tr>
<tr>
<td>middleName</td>
<td>A string.</td>
</tr>
<tr>
<td>mobilePhoneNumber</td>
<td>A string.</td>
</tr>
<tr>
<td>nickName</td>
<td>A string.</td>
</tr>
<tr>
<td>office</td>
<td>A string.</td>
</tr>
<tr>
<td>otherAddress</td>
<td>A structure with the following fields: Street, City, State, Zip, Country.</td>
</tr>
<tr>
<td>otherPhoneNumber</td>
<td>A string.</td>
</tr>
<tr>
<td>pager</td>
<td>A string.</td>
</tr>
</tbody>
</table>
The `cfexchangemail` tag filters can have the following name attributes and associated value, or `to` and `from` attributes that you use to specify the filter parameters for the specified action:

<table>
<thead>
<tr>
<th>name attribute</th>
<th>Specification attributes</th>
<th>Specification attribute values</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxRows</td>
<td>value</td>
<td>A positive integer that specifies the maximum number of matching rows to return. By default, the maximum number of rows is 100. You can also specify -1; to return all matching rows.</td>
</tr>
<tr>
<td>bcc</td>
<td>value</td>
<td>A comma-delimited list of Exchange or web e-mail addresses.</td>
</tr>
<tr>
<td>cc</td>
<td>value</td>
<td>A comma-delimited list of Exchange or web e-mail addresses.</td>
</tr>
<tr>
<td>folder</td>
<td>value</td>
<td>The forward slash delimited path from the root of the Exchange mailbox to the folder to search. By default, the filter searches the top level of the Inbox. The <code>cfexchangemail</code> tag searches only the specified folder, and does not search any subfolders. If a folder name contains a forward slash, use the <code>xF8FF</code> escape sequence to specify the character in the name. For the <code>get</code> and <code>move</code> actions, you can use the <code>cfexchangemail</code> tag <code>folder</code> attribute instead of this field; however, this field takes precedence over the value specified in the <code>folder</code> attribute.</td>
</tr>
<tr>
<td>fromID</td>
<td>value</td>
<td>An Exchange or web e-mail address.</td>
</tr>
<tr>
<td>hasAttachment</td>
<td>value</td>
<td>A Boolean value</td>
</tr>
<tr>
<td>Attribute</td>
<td>Value Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>importance</td>
<td>value</td>
<td>One of the following values: high, normal, low</td>
</tr>
<tr>
<td>isRead</td>
<td>value</td>
<td>A Boolean value.</td>
</tr>
<tr>
<td>message</td>
<td>value</td>
<td>A string.</td>
</tr>
<tr>
<td>MessageType</td>
<td>value</td>
<td>One of the following values: Mail, Meeting, Meeting_Cancel, Meeting_Request, Meeting_Response, or All. If you omit this attribute, the filter gets messages of all types. The Meeting attribute gets messages with Meeting_Cancel, Meeting_Request, and Meeting_Response types.</td>
</tr>
<tr>
<td>MeetingUID</td>
<td>value</td>
<td>A case-sensitive Exchange calendar event UID. Meeting UIDs are used in Meeting_request or Meeting_response message types only. Do not specify this field if you specify a MessageType field value of Mail.</td>
</tr>
<tr>
<td>sensitivity</td>
<td>value</td>
<td>One of the following values: normal, personal, private, confidential</td>
</tr>
<tr>
<td>subject</td>
<td>value</td>
<td>A string.</td>
</tr>
<tr>
<td>timeReceived</td>
<td>from to</td>
<td>A string that ColdFusion can interpret as a date-time value.</td>
</tr>
<tr>
<td>timeSent</td>
<td>from to</td>
<td>A string that ColdFusion can interpret as a date-time value.</td>
</tr>
<tr>
<td>toID</td>
<td>value</td>
<td>A comma-delimited list of Exchange or web e-mail addresses.</td>
</tr>
<tr>
<td>uid</td>
<td>value</td>
<td>A case-sensitive Exchange message UID.</td>
</tr>
</tbody>
</table>

The cfexchangetask tag filters can have the following name attributes and associated value, or to and from at
attributes that you use to specify the filter parameters for the specified action:

<table>
<thead>
<tr>
<th>name attribute</th>
<th>Specification attributes</th>
<th>Specification attribute values</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxRows</td>
<td>value</td>
<td>A positive integer specifying the maximum number of matching rows to return. By default, the maximum number of rows is 100. You can also specify -1; to return all matching rows.</td>
</tr>
<tr>
<td>actualWork</td>
<td>value</td>
<td>A number representing the number of hours. Use decimal numbers to specify minutes.</td>
</tr>
<tr>
<td>billingInfo</td>
<td>value</td>
<td>A string.</td>
</tr>
<tr>
<td>companies</td>
<td>value</td>
<td>A string.</td>
</tr>
<tr>
<td>dateCompleted</td>
<td>value</td>
<td>A string that ColdFusion can interpret as a date-time value.</td>
</tr>
<tr>
<td>dueDate</td>
<td>from to</td>
<td>A string that ColdFusion can interpret as a date-time value.</td>
</tr>
<tr>
<td>mail_ID</td>
<td>value</td>
<td>A comma-delimited list of Exchange mail IDs. This filter value is useful if the connection user has delegate rights for multiple users and you want to select the tasks of a limited number of those users.</td>
</tr>
<tr>
<td>message</td>
<td>value</td>
<td>A string.</td>
</tr>
<tr>
<td>mileage</td>
<td>value</td>
<td>A string.</td>
</tr>
<tr>
<td>percentCompleted</td>
<td>value</td>
<td>A number between 0 and 100.</td>
</tr>
</tbody>
</table>
| priority          | value                    | One of the following values:  
  - high  
  - normal  
  - low  
| reminderDate      | value                    | A string that ColdFusion can interpret as a date-time value. |
| startDate         | from to                  | A string that ColdFusion can interpret as a date-time value. |
The `cfexchangefilter` tag specifies the conditions to match when ColdFusion gets mail messages, calendar entries, tasks, or contacts. Only those entries that match the specified filter conditions are returned in the structure specified by the parent tag's `name` attribute. If the filter specifies a field that takes a text string, such as `Message` and `subject` or `Subject`, ColdFusion returns items that contain the exact phrase that you specify in the `value` attribute.

The `cfexchangefilter` tag must be a child tag of a `cfexchangemail`, `cfexchangecalendar`, `cfexchangecontact`, or `cfexchangetask` tag with an `action` attribute value of `get`. If you specify multiple `cfexchangefilter` tags in the body of a ColdFusion exchange tag, such as `cfexchangemail`, the specified filters are cumulative, and the selected records match the conditions specified in all the `cfexchangefilter` tags. If you specify multiple `cfexchangefilter` tags with the same `name` attribute value, the last tag with that attribute specifies the filter conditions.

**Example**

The following example gets the mail messages that were sent to a user during the last week from any e-mail address that includes `adobe.com`. To focus on getting messages, rather than on displaying data, the example uses the `cfdump` tag to show the results.

```cfc
cfset endTime = Now()>
cfset startTime = DateAdd("d","-7", endTime)>
cfexchangemail action="get" name="weeksMail" server="#exchangeServerIP#" username="#user1#" password="#password1#">
  <cfexchangefilter name="FromID" value="adobe.com">
  <cfexchangefilter name="TimeSent" from="#startTime#" to="#endTime#">
</cfexchangemail>
<cfdump var="#weeksMail#">
```
**cfexchangefolder**

**Description**

Lets you perform various actions on the mail folder, such as get folder information, find folders, or create, copy, modify, move, delete, and empty the contents of a folder.

**History**

ColdFusion 10: Added this tag.

**Category**

Communications tags

**Syntax**

```xml
getExtendedInfo
<cfexchangefolder
  action = "getExtendedInfo"
  folderID = "Exchange folder UID"
  connection = "connection_ID"
  name = "query_name"/>
OR

gAcceptInfo
<cfexchangefolder
  action = "getInfo"
  connection = "connection_ID"
  folderID = "Exchange folder UID"
  name = "query_name"/>
OR

gAcceptInfo
<cfexchangefolder
  action = "getInfo"
  folderPath = "Exchange_folder_Name"
  connection = "connection_ID"
  name = "query_name" pathDelimeter = "delimiter_characters"/>

findSubFolders
<cfexchangefolder
  action = "findSubFolders"
  folderID = "Exchange folder UID"
  connection = "connection_ID"
  name = "query_name"/>

create
<cfexchangefolder
  action = "create"
  folder = "struct"
  parentFolderID = "folder_UID"
  connection = "connection_ID"
```
result = "variable for contact UID"/>
copy
<cfexchangefolder
    action = "copy"
    destinationFolderID = "Folder_UID"
    sourceFolderID = "folder_UID">
    connection = "connection_ID"
    result = "variable for contact UID"/>
delete
<cfexchangefolder
    action = "delete"
    deleteType = "hardDelete|softDelete|moveToDeletedItems"
    uid = "folder UID">
    connection = "connection_ID"/>
move
<cfexchangefolder
    action = "move"
    destinationFolderID = "Folder_UID"
    sourceFolderID = "folder_UID">
    connection = "connection_ID"
    result = "variable for contact UID"/>
modify
<cfexchangefolder
    action = "modify"
    uid = "folder UID"
    folder = "strut"
    connection = "connection_ID"/>
empty
<cfexchangefolder
    action = "empty"
    uid = "folder UID"
    deleteType = "hardDelete|softDelete|moveToDeletedItems">
deleteSubFolder = "true|false"
connection = "connection_ID"/

See also
cfexchangecalendar, cfexchangeconnection, cfexchangefilter, cfexchangemail, cfexchangetask,
Interacting with Microsoft Exchange Servers in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>N/A</td>
<td>Required</td>
<td></td>
<td>The action to take. Must be one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getInfo</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getExtendedInfo</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• findSubFolder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• create</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• copy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• move</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• modify</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• empty</td>
</tr>
<tr>
<td>uid</td>
<td>getExtendedInfo / getInfo</td>
<td>Not required for get ExtendedInfo and getInfo, if folder path is specified.</td>
<td>UID that is used to identify the folder in which the actions are performed.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>getExtendedInfo / getInfo / findSubFolders</td>
<td>Required</td>
<td></td>
<td>The name of the ColdFusion query variable that contains the returned information about the folder.</td>
</tr>
<tr>
<td>folderID</td>
<td>getExtendedInfo / getInfo / findSubFolders / delete / modify / empty</td>
<td>Not required for get ExtendedInfo and getInfo, if folder path is specified.</td>
<td>UID that is used to identify the folder in which the actions are performed.</td>
<td></td>
</tr>
<tr>
<td><strong>FolderPath</strong></td>
<td><strong>getExtendedInfo / getInfo</strong></td>
<td><strong>Not required if folderID is specified.</strong></td>
<td><strong>Full path to the folder where the action has to be performed. If you do not specify the path delimiter, / is taken by default.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>pathDelimiter</strong></td>
<td><strong>getExtendedInfo / getInfo</strong></td>
<td><strong>Optional /</strong></td>
<td><strong>Lets you specify the delimiter that is used to separate the folders.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>parentFolderId</strong></td>
<td><strong>create</strong></td>
<td><strong>Required</strong></td>
<td><strong>UID that is used to identify the folder in which you create subfolders.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>connection</strong></td>
<td><strong>All actions</strong></td>
<td><strong>Required</strong></td>
<td><strong>The name of the connection to the Exchange server, as specified in the cfe exchangeconnection tag. If you omit this attribute, create a temporary connection by specifying cfexchangeconnection tag connection open action attributes in the cfexchangecontact tag.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>result</strong></td>
<td><strong>create / copy / move</strong></td>
<td><strong>Required</strong></td>
<td><strong>A query variable that contains the result returned from the exchange server when one of the action is performed.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>destinationFolderId</strong></td>
<td><strong>copy / move</strong></td>
<td><strong>Required</strong></td>
<td><strong>A case-sensitive Exchange UID value that uniquely identifies the destination folder.</strong></td>
<td></td>
</tr>
<tr>
<td>sourceFolderID</td>
<td>copy / move</td>
<td>Required</td>
<td>The UID that is used to identify the folder from which you copy or move folders to the destination folder.</td>
<td></td>
</tr>
<tr>
<td>deleteType</td>
<td>delete / move</td>
<td>Optional</td>
<td>moveToDeletedItems</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• hardDelete: Removes a folder permanently from the Exchange server.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• softDelete: Moves a folder to the dumpster in Exchange server, if dumpster is enabled.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• moveToDeletedItems: Moves a folder to the deleted items folder.</td>
<td></td>
</tr>
<tr>
<td>deleteSubFolder</td>
<td>empty</td>
<td>Optional</td>
<td>False</td>
<td>If true, deletes the subfolder.</td>
</tr>
<tr>
<td>folder</td>
<td>create / modify</td>
<td>Required</td>
<td>A struct that contains the required information of the folder that has to be created or modified, such as display name and folder class.</td>
<td></td>
</tr>
</tbody>
</table>

Result struct values for cffilefolder action = "getExtendedInfo"

For the action getExtendedInfo, the result is a struct that contains the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChildFolderCount</td>
<td>The number of child folders that the parent folder has.</td>
</tr>
<tr>
<td>displayName</td>
<td>The display name of the folder.</td>
</tr>
<tr>
<td>folderClass</td>
<td>The folder class.</td>
</tr>
<tr>
<td>FolderPath</td>
<td>Path to the exchange server folder.</td>
</tr>
</tbody>
</table>
ManagedFolder

<table>
<thead>
<tr>
<th>Struct that contains the following properties:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• canDelete: Boolean value that indicates if the user can delete objects in the folder.</td>
</tr>
<tr>
<td>• canRenameOrMove: Boolean value that indicates if the user can rename or move objects in the folder.</td>
</tr>
<tr>
<td>• mustDisplayComment: Boolean value that indicates if the client application must display the comment property to the user.</td>
</tr>
<tr>
<td>• HasQouta: Boolean value that indicates if the folder has a quota.</td>
</tr>
<tr>
<td>• IsManagedFoldersRoot: Boolean value that indicates if the folder is the root of the managed folder hierarchy.</td>
</tr>
<tr>
<td>• ManagedFolderId: The ID of the managed folder.</td>
</tr>
<tr>
<td>• comment: The comment that is associated with the folder.</td>
</tr>
<tr>
<td>• StorageQuota: The storage quota of the folder.</td>
</tr>
<tr>
<td>• FolderSize: The size of the folder.</td>
</tr>
<tr>
<td>• HomePage: The home page associated with the folder.</td>
</tr>
</tbody>
</table>

| mustDisplayComment |
| Boolean value that indicates if the client application must display the comment property to the user. |

| ParentFolderId |
| The ID of the folder’s parent folder. |

| Permission |
| List of permissions for the folder. This is an array of permission structs with the following values: |
- **canCreateItems**: Boolean value that indicates if the user can create new items.
- **canCreateSubFolders**: Boolean value that indicates if the user can create sub-folders.
- **deleteItems**: Indicates if/how the user can delete existing items. The values are None (the user does not have the associated permission), Owned (the user has the associated permission on items that it owns), and All (the user has the associated permission on all items).
- **displayPermissionLevel**: The permission level that Outlook displays for this folder permission. It can have the following values: None, Owner, PublishingEditor, Editor, PublishingAuthor, FreeBusyTimeAndSubjectAndLocation, FreebusyTimeOnly, Author, NonEditingAuthor, Reviewer, Contributor, and Custom.
- **EditItems**: Indicates the items in a folder that the user has permission to edit. Values are none, owned, or all.
- **isFolderContact**: Boolean value that indicates if the user is a contact for the folder.
- **isFolderOwner**: Boolean value that indicates if the user owns the folder.
- **isFolderVisible**: Boolean value that indicates if the folder is visible to the user.
- **PermissionLevel**: Represents the combination of permissions that a user has on a folder. The values are same as that of `displayPermissionLevel` except `FreeBusyTimeAndSubjectAndLocation` and `FreebusyTimeOnly`.
- **readItems**: The read item access permission. The values are as follows: None (user has no read access on the items in the folder), TimeOnly (user can read the start and end date and time of appointments; applies only to Calendar folders), TimeAndSubjectAndLocation (user can read the start and end date and time, subject, and location of appointments; applies only to calendar folders), and FullDetails (user has access to full details of items).
- **UserIDDisplayName**: Display name of the user.
- **UserIDprimarySMTPAddress**: The primary SMTP address of the user.
- **userIDSID**: SID of the user.
- **UserIDstandardUser**: Indicates the standard user, if default (the default delegate user, used to define default delegate permissions) and Anonymous (the anonymous delegate user, used to define delegate permissions for unauthenticated users).
Result struct values for cffile action = "getInfo"

For the action getInfo, the result is a struct that contains the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChildFolderCount</td>
<td>The number of child folders this folder has</td>
</tr>
<tr>
<td>displayName</td>
<td>The display name of the folder</td>
</tr>
<tr>
<td>folderClass</td>
<td>Folder class</td>
</tr>
<tr>
<td>folderPath</td>
<td>Path to the exchange server folder.</td>
</tr>
<tr>
<td>UID</td>
<td>ID of the folder</td>
</tr>
<tr>
<td>ParentFolderId</td>
<td>The Id of the current folder's parent folder</td>
</tr>
<tr>
<td>TotalCount</td>
<td>Total number of items contained in the folder</td>
</tr>
<tr>
<td>UnreadCount</td>
<td>The number of unread items in the folder</td>
</tr>
</tbody>
</table>

Filter parameters for cffile action = "findSubFolders"

For the action findSubFolders, the result is a query that contains details of the subfolders. The values are same as that of cffile action = "getInfo".

Example

The following code shows how you can perform the actions getExtendedInfo, findSubFolders, getInfo, copy, delete, modify, move, and create.

```xml
<cfexchangeconnection action="open" username="folder" password="Password" server="IP_Address" serverversion="2010" connection="conn1">
<cffile action="getExtendedInfo" connection="conn1" name="result" folderpath="Drafts">
<cffile action="getExtendedInfo" connection="conn1" name="result3" folderpath="Inbox">
<!--- Checking if the folder is already present.. If it is present then delete it --->
<cffile action="findsubfolders" connection="conn1" name="final" uid="#result3.uid#">
<cffilefilter name="displayname" value="folder1">
</cffilefilter>
</cfexchangefolder>
<cffile action="findsubfolders" connection="conn1" name="final1" uid="#result.uid#">
```
<cfexchangefilter name="displayname" value="folder1">
</cfexchangefilter>
<cfif (#final.displayname# EQ "folder1")>
  <cfexchangefolder action="delete" connection="conn1" uid="#final.uid#"
deletetype="harddelete">
    <cfscript>
      sleep(1000);
    </cfscript>
  </cfif>
<cfelseif (#final1.displayname# EQ "folder1")>
  <cfexchangefolder action="delete" connection="conn1" uid="#final1.uid#"
deletetype="harddelete">
    <cfscript>
      sleep(1000);
    </cfscript>
  </cfif>
<cfset newfolder = structnew()>
<cfset newfolder.displayname = "folder1">
<cfset newfolder.folderclass = "IPF.Note">
<cfexchangefolder action="create" connection="conn1" parentfolderid="#result.uid#" folder="#newfolder#" result="result1">
</cfexchangefolder>
<!--- Source folder --->
<cfexchangefolder action="findsubfolders" connection="conn1" name="result2" uid="#result.uid#">
<cfexchangefilter name="displayname" value="folder1">
</cfexchangefolder>
<!---modifying the folder--->
<cfset modfolder = structnew()>
<cfset modfolder.displayname = "exchange">
<cfset modfolder.folderclass = "IPF.Calendar">
<cfexchangefolder action="modify" connection="conn1" uid="#result2.uid#" folder="#modfolder#">
</cfexchangefolder>
<cfexchangefolder action="findsubfolders" connection="conn1" name="result5" uid="#result.uid#">
<cfexchangefilter name="displayname" value="exchange">
</cfexchangefolder>
<cfdump var="#result5#">
<!--- Destination folder --->
<cfexchangefolder action="getInfo" connection="conn1" name="result3" folderpath="Inbox">
<cfexchangefolder action="copy" connection="conn1" result="copiedfolderid" sourcefolderid="#result2.uid#" destinationfolderid="#result3.uid#">
<cfexchangefolder action="findsubfolders" connection="conn1" name="result2" uid="#result.uid#">
<cfexchangefilter name="displayname" value="exchange">
</cfexchangefolder>
<cfdump var="#result5#">
<cfexchangefolder action="findsubfolders" connection="conn1" name="result4" uid="#result3.uid#">
<cfexchangefilter name="displayname" value="exchange">
</cfexchangefolder>
<cfdump var="#result4#">
<!---Deleting the folder from source --->
<cfexchangefolder action="delete" connection="conn1" uid="#result2.uid#" deletetype="harddelete">
</cfexchangefolder>
<cfexchangefolder action="move" connection="conn1" result="movedfolderid" sourcefolderid="#result4.uid#" destinationfolderid="#result.uid#"
<cfexchange folder action="findsubfolders" connection="conn1" name="result6"
uid="#result.uid#">
  <cfexchange filter name="displayname" value="exchange"/>
</cfexchange folder>
<cfdump var="#result6#"/>
<cfexchangefolder action="delete" connection="conn1" uid="#result6.uid#"
deletetype="harddelete">
**cfexchangemail**

**Description**

 Gets mail messages and attachments, deletes messages, and sets properties for messages on a Microsoft Exchange server.

**History**

ColdFusion 10: Added the attribute `serverVersion` and `folderID`.

ColdFusion 8: Added this tag.

**Category**

*Communications tags*

**Syntax**

```xml
delete
cfexchangemail
    required
    action = "delete"
    uid = "message UID,message UID,..."
    optional
    connection = "connection ID"
    folder = "Exchange folder path">

deleteAttachments
cfexchangemail
    required
    action = "deleteAttachments"
    uid = "message UID"
    optional
    connection = "connection ID"
    folder = "Exchange folder path">

get
cfexchangemail
    required
    action = "get"
    name = "query identifier"
    optional
    connection = "connection ID"
    folder = "Exchange folder path"
    folderID = "Exchange folder UID"

    <cfexchangefilter name = "filter type" value = "filter value">
    <cfexchangefilter name = "filter type" value = "filter value">
    ...
</cfexchangemail>

getAttachments
cfexchangemail
    required
    action = "getAttachments"
    name = "query identifier"```
uid = "message UID"
optional
attachmentPath = "directory path"
connection = "connection ID"
folder = "Exchange folder path"
generateUniqueFilenames = "no|yes">

getMeetingInfo
<cfexchangeml
required
action = "getMeetingInfo"
meetingUID = "meeting UID"
name = "query identifier"
optional
connection = "connection ID"
mailUID = "message UID">

move
<cfexchangeml
required
action = "move"
destinationFolder = "Exchange folder path"
optional
connection = "connection ID"
folder = "Exchange folder path">
folderID = "Exchange folder UID"

<cfexchangefilter name = "filter type" value = "filter value">
<cfexchangefilter name = "filter type" value = "filter value">
...
</cfexchangeml>

set
<cfexchangeml
required
action = "set"
message = "#structure with values to set#"
uid = "message UID">
optional
connection = "connection ID"
(folder = "Exchange folder path")
(folderID = "Exchange folder UID")

⚠️ Note

If you omit the connection attribute, create a temporary connection by specifying `cfexchange` connection tag attributes in the `cfexchangemail` tag. In this case, ColdFusion closes the connection when the tag completes. For details, see the `cfexchangeconnection` tag open action.

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfexchangecalendar, cfexchangeconnection, cfexchangecontact, cfexchangefilter, cfexchange

task, Interacting with Microsoft Exchange Servers in the Developing ColdFusion Applications

Attributes

⚠️ Note

If an attribute, such as folder or destinationFolder takes a folder path, and the folder name contains forward slashes, specify the folder name by using the `xF8FF` escape character to prevent exchange from interpreting the character as a path delimiter.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>all</td>
<td>Required</td>
<td></td>
<td>The action to take. Must be one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• deleteAttachments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• get</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getAttachment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getMeetingInfo</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• move</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• set</td>
</tr>
<tr>
<td>Parameter</td>
<td>Action</td>
<td>Default</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>attachmentPath</td>
<td>getAttachments</td>
<td>Optional</td>
<td>The filepath of the directory in which to put the attachments. If the directory does not exist, ColdFusion creates it. Note: If you omit this attribute, ColdFusion does not save any attachments. If you specify a relative path, the path root is the ColdFusion temporary directory, which is returned by the <code>GetTempDirectory</code> function.</td>
<td></td>
</tr>
<tr>
<td>connection</td>
<td>all</td>
<td>Optional</td>
<td>The name of the connection to the Exchange server, as specified in the <code>cfxchangeconnection</code> tag. If you omit this attribute, create a temporary connection by specifying <code>cfexchangeconnection</code> tag open action attributes in the <code>cfexchangecalendar</code> tag.</td>
<td></td>
</tr>
<tr>
<td>destinationFolder</td>
<td>move</td>
<td>Required</td>
<td>The forward slash delimited path, relative to the root of the mailbox, of the folder to which to move the message or messages.</td>
<td></td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>folder</strong></td>
<td>all except getMeetingInfo</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>The forward slash delimited path, relative to the root of the mailbox, of the folder that contains the message or messages. The cfexchangemail tag looks in the specified folder only, and does not search subfolders. For the get and move actions specifying a cfexcchangefilter child tag with a name=&quot;folder&quot; attribute is equivalent to setting this attribute, and takes precedence over this attribute's value. If you omit this attribute, or for get and move actions, if you do not use the corresponding cfexchangefilter setting, Exchange looks in the top level of the Inbox.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>folderID</strong></td>
<td>__get, move, set</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>A case-sensitive Exchange UID value that uniquely identifies the folder. If not specified, folder is used. If either folder or folderID are not specified, the inbox is used as the default folder to perform the operation.</td>
<td></td>
</tr>
<tr>
<td>generateUniqueFilenames</td>
<td>getAttachments</td>
<td>Optional</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>mailUID</td>
<td>getMeetingInfo</td>
<td>Optional</td>
</tr>
<tr>
<td>meetingUID</td>
<td>getMeetingInfo</td>
<td>Required</td>
</tr>
<tr>
<td>attribute</td>
<td>value</td>
<td>type</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>---------</td>
</tr>
<tr>
<td>message</td>
<td>set</td>
<td>Required</td>
</tr>
<tr>
<td>name</td>
<td>getAttachments</td>
<td>Required</td>
</tr>
<tr>
<td>serverVersion</td>
<td>Optional</td>
<td>2007</td>
</tr>
</tbody>
</table>
### Usage

The `cfexchangemail` tag performs mail actions on an Exchange server that you cannot do by using the `cfmail` tag. (You must use the `cfmail` tag to send, forward, and reply to mail messages.) Use the `cfexchangemail` tag to perform the following actions:

- Permanently delete one or more mail messages from the server.
- Get the attachments for a specific message.
- Get one or more messages that conform to an optional set of filter specifications, such as the subject, sender or recipient ID, time received, and so on.
- Get the attachments for a specific message.
- Get detailed information about a meeting for which you have a notification, such as a meeting request or cancellation notice.
- Move one or more messages from one folder to another, including to the Deleted Items folder.
- Set the properties of a specific mail message.

To use this tag, you must have a connection to an Exchange server. If you are using multiple tags that interact with the exchange server, such as if you are creating several contact records, use the `cfexchangeconnection` tag to create a persistent connection. You then specify the connection identifier in each `cfexchangemail` tag, or any other ColdFusion Exchange tag, if you are also accessing tasks, contacts, or connections. Doing this saves the overhead of creating and closing the connection for each tag. Alternatively, you can create a temporary connection that lasts only for the time that ColdFusion processes the single `cfexchangemail` tag. To do this, you specify the connection attributes directly in the `cfexchangemail` tag. For details on the connection attributes, see the `cfexchangeconnection` tag.

#### The delete action

The `delete` action permanently deletes a message from the server, and is equivalent to the Outlook Shift-Delete keystroke action. Use the `move` action to move a message to the Deleted Items folder, which is equivalent to the Outlook Delete keystroke action. When you specify the `delete` action you must specify a `uid` attribute with a comma-delimited list of one or more Exchange UIDs that identify the tasks that you want to delete. You can use the `get` action, with an appropriate filter expression, to determine the UID values to specify. If all UIDs that you specify are invalid, the `cfexchangemail` tag generates an error. If at least one UID is valid, the tag ignores any invalid UIDs and deletes the items specified by the valid UID.

#### The get action
When you specify the get action, you use child `cfexchangefilter` tags to specify the messages to get. For detailed information, see `cfexchangefilter`. When the tag completes processing, the query object specified by the `name` attribute contains one record for each matching message that was found. Each record has the following columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCC</td>
<td>A comma-delimited list of Exchange user IDs or web e-mail.</td>
</tr>
<tr>
<td>CC</td>
<td>A comma-delimited list of Exchange user IDs or web e-mail addresses.</td>
</tr>
<tr>
<td>Folder</td>
<td>The forward slash delimited path from the root of the Exchange mailbox to the mail folder containing the message.</td>
</tr>
<tr>
<td>FromID</td>
<td>An Exchange user IDs or web e-mail addresses.</td>
</tr>
<tr>
<td>HasAttachment</td>
<td>A Boolean value that indicates whether the message has at least one attachment.</td>
</tr>
<tr>
<td>HTMLMessage</td>
<td>A string containing a HTML-formatted version of the message.</td>
</tr>
<tr>
<td>IsRead</td>
<td>A Boolean value.</td>
</tr>
<tr>
<td>Message</td>
<td>A string with a plain-text version of the message contents.</td>
</tr>
<tr>
<td>MessageType</td>
<td>One of the following strings:</td>
</tr>
<tr>
<td></td>
<td>• Mail</td>
</tr>
<tr>
<td></td>
<td>• Meeting_Cancel</td>
</tr>
<tr>
<td></td>
<td>• Meeting_Request</td>
</tr>
<tr>
<td></td>
<td>• Meeting_Response</td>
</tr>
<tr>
<td>MeetingResponse</td>
<td>If the message type is Meeting_response, this column contains the response code as one of the following strings: Accept, Decline, or Tentative.</td>
</tr>
<tr>
<td></td>
<td>This field is not used for other message types.</td>
</tr>
<tr>
<td>MeetingUID</td>
<td>If the message type is Meeting_Cancel, Meeting_request, or Meeting_response this column contains the UID of the calendar event for which this message was sent. Use this value in the <code>cfexchangecalendar</code> tag to respond to a request. This field is not used for the Mail message type.</td>
</tr>
</tbody>
</table>
| **Sensitivity** | One of the following strings:  
• public  
• private  
• normal  
• company-confidential |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subject</strong></td>
<td>A string.</td>
</tr>
<tr>
<td><strong>TimeReceived</strong></td>
<td>A ColdFusion date-time object.</td>
</tr>
<tr>
<td><strong>TimeSent</strong></td>
<td>A Coldfusion date-time object.</td>
</tr>
<tr>
<td><strong>Told</strong></td>
<td>A comma-delimited list of Exchange user IDs or web mail IDs.</td>
</tr>
<tr>
<td><strong>UID</strong></td>
<td>The Exchange UID of the message.</td>
</tr>
</tbody>
</table>

**Note**

An invitation sender can get a meeting request message only if the sender is on the attendee list.

The getAttachments action

When you use the `getAttachments` action, specify a single UID and a `name` attribute. The `cfexchangecontact` tag populates a query object specified by the `name` attribute with one record for each attachment. Each record has the following information about the mail attachment specified by the UID:

<table>
<thead>
<tr>
<th><strong>Column name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>attachmentFileName</td>
<td>The filename of the attachment.</td>
</tr>
</tbody>
</table>
| attachmentFilePath    | The absolute path of the attachment file on the server.  
If you omit the `attachmentPath` attribute, this column contains the empty string. |
| CID                   | The content-ID of the attachment. Used in HTML `img` tags to embed images in a message. |
| mimeType              | The MIME type of the attachment, such as text/html.                           |
| isMessage             | A Boolean value that specifies whether the attachment is a message.           |
| size                  | The attachment size in bytes.                                                |

The tag places the attachments in the directory specified by the `attachmentPath` attribute. If you omit the `attachmentPath` attribute, ColdFusion does not get any attachments; it gets the information about the attachments. This lets you determine the attachments without incurring the overhead of getting the attachment files.
If a message has multiple attachments with the same name, the attachment information structure always lists the attachments with their original, duplicate, names, even if you specify `generateUniqueFilenames="yes"`. The `generateUniqueFilenames` attribute only affects the names of the files on disk. Use the following syntax to specify an in-memory `attachmentPath` directory. In-memory files are not written to disk and speed processing of transient data.

```
attachmentpath = "ram:///path"
```

The path can include multiple directories, for example `ram:///petStore/orders/messageAttachments`. Create all directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications. The `getAttachments` action works only if authentication for EWS (Exchange Web Services) is set to basic in the server setup of Exchange. IWA (Integrated Windows Authentication) is not supported.

**The `getMeetingInfo` action**

You use the `getMeetingInfo` action to get meeting-specific information, such as the meeting start and end times, location, and so on, about a meeting for which you have received a notification message, such as an invitation request or cancellation notice. This information is not available directly in the notification message query object that is returned by the `get` action.

⚠️ **Note**

At the time of publication, the following information does not completely reflect the behavior of the `getMeetingInfo` action. For updated information, see `cfexchangemail` in the ColdFusion documentation available online in HTML on the Adobe website.

When you specify the `getMeetingInfo` action, you specify a `meetingUID` attribute with the UID of the meeting. You get this UID value from the query record that is returned by the `get` action. You can optionally specify a `messageUID` attribute with the UID of the specific message that contains the notification; if you receive multiple messages about a single meeting, you can use this attribute to select a single notification message. When the tag completes processing, the query object specified by the `name` attribute contains one record for each matching message that was found. Each record has the following columns:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllDayEvent</td>
<td>A Boolean value that indicates whether this is an all day event.</td>
</tr>
<tr>
<td>Duration</td>
<td>The duration of the event in minutes.</td>
</tr>
<tr>
<td>EndTime</td>
<td>The end time of the event, in ColdFusion ODBC date-time format.</td>
</tr>
<tr>
<td>From</td>
<td>The mail ID of the person who sent the meeting notification.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HasAttachment</td>
<td>A Boolean value that indicates whether this event has attachments.</td>
</tr>
<tr>
<td>Importance</td>
<td>One of the following values:</td>
</tr>
<tr>
<td></td>
<td>• high</td>
</tr>
<tr>
<td></td>
<td>• normal</td>
</tr>
<tr>
<td></td>
<td>• low</td>
</tr>
<tr>
<td>IsRecurring</td>
<td>A Boolean value that indicates whether this event repeats.</td>
</tr>
<tr>
<td>Location</td>
<td>A string that specifies the location of the event.</td>
</tr>
<tr>
<td>MeetingUID</td>
<td>The UID of the event in the calendar.</td>
</tr>
<tr>
<td>Message</td>
<td>A string that contains a message about the event.</td>
</tr>
<tr>
<td>OptionalAttendees</td>
<td>A comma-delimited list of mail IDs.</td>
</tr>
<tr>
<td>Organizer</td>
<td>A string. This value is not necessarily an Exchange ID or e-mail address.</td>
</tr>
<tr>
<td>Reminder</td>
<td>The time, in minutes before the event, at which to display a reminder message.</td>
</tr>
<tr>
<td>RequiredAttendees</td>
<td>A comma-delimited list of mail IDs.</td>
</tr>
<tr>
<td>Resources</td>
<td>A comma-delimited list of mail IDs for Exchange scheduling resources, such as conference rooms and display equipment.</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>One of the following values:</td>
</tr>
<tr>
<td></td>
<td>• normal</td>
</tr>
<tr>
<td></td>
<td>• company-confidential</td>
</tr>
<tr>
<td></td>
<td>• personal</td>
</tr>
<tr>
<td></td>
<td>• private</td>
</tr>
<tr>
<td>StartTime</td>
<td>The start time of the event, in ODBC date-time format.</td>
</tr>
<tr>
<td>Subject</td>
<td>A string describing the event subject.</td>
</tr>
<tr>
<td>TimeReceived</td>
<td>The time the message was received, in ODBC date-time format.</td>
</tr>
<tr>
<td>UID</td>
<td>The UID of the message that contains the event notification.</td>
</tr>
</tbody>
</table>

The move action
Use the move action to move one or more messages from one folder to another folder. You can use this action to move messages to the Deleted Items folder, which is equivalent to the Outlook Delete keystroke action. When you specify the move action you specify the destination folder, and optionally the folder containing the messages to move. (The default source folder is the Inbox). You use child cfexchangefilter tags to specify the messages to get. For detailed information, see cfexchangefilter.

The set action

When you specify the set action, the structure specified by the message attribute contains key-value pairs that specify the message properties to set. The following table lists the key names and their valid values:

<table>
<thead>
<tr>
<th>Key name</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsRead</td>
<td>yes, no</td>
</tr>
<tr>
<td>Importance</td>
<td>high, normal, low</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>normal, company-confidential, personal, private</td>
</tr>
</tbody>
</table>

Example

The following example gets the attachments to all mail messages in the Inbox from docuser2 in the last week. It puts each message's attachments in a directory with a unique name. It cannot use the UID as a filename because, for each message with attachments, the UID can contain the application reports of the UID, directory path, subject, date, and sender of the message, followed by a table that lists the message's attachments. The table includes the attachment name, size, and MIME type.
<!--- Index for message attachment directory --->
<cfset i=1>
<!--- Dates for date range --->
<cfset rightNow = Now()>
<cfset lastWeek = DateAdd("d","-7", rightNow)>

<cfexchangeconnection
action="open"
username="#user1#"
password="#password1#"
server="#exchangeServerIP#"
connection="testconn1">
<cfexchangemail action="get" folder="Inbox" name="weeksMail"
connection="testconn1">
<cfexchangefilter name="FromID" value="docuser2">
<cfexchangefilter name="TimeSent" from="#lastWeek#" to="#rightNow#">
</cfexchangemail>

<cfloop query="weeksMail">
<cfif weeksMail.HasAttachment>
<cfexchangeconnection
action="open"
username="#user1#"
password="#password1#"
server="#exchangeServerIP#"
connection="testconn1">
<cfexchangemail action="getAttachments"
connection="testconn1"
folder="Inbox\MailTest"
uid="#weeksMail.uid#"
name="attachData"
attachmentPath="C:\temp\cf_files\attachments\msg_#i#"
generateUniqueFilenames="yes">
<cfoutput>
Message ID #weeksMail.uid# attachments are in the directory
C:\temp\cf_files\attachments\Msg_#i#<br />
<br />
Message information:<br />
Subject: #weeksMail.Subject#<br />
Sent: #dateFormat(weeksMail.TimeSent)#<br />
From: #weeksMail.FromID#<br />
<br />
Attachments<br />
<cftable query="attachData" colheaders="yes">
<cfcol header="File Name" text="#attachmentFilename#">
<cfcol header="Size" text="#size#">
<cfcol header="MIME type" text="#mimeType#">
</cftable>
</cfoutput>
</cfset i++>
</cfif>
</cfloop>
<cfexchangeconnection action="close" connection="testconn1">
cfexchangetask

Description

Creates, deletes, modifies, and gets Microsoft Exchange tasks, and gets task attachments.

⚠️ Note

For all actions, see cfexchangeconnection for additional attributes that you use if you do not specify the connection attribute.

History

ColdFusion 10: Added the attribute serverVersion.ColdFusion 8: Added this tag.

Category

Communications tags

Syntax
create
<cfexchangetask
  required
  action = "create"
  task = "#task information structure#"
optional
  connection = "connection ID"
  result = "variable for event UID">

delete
<cfexchangetask
  required
  action = "delete"
  uid = "task UID,task UID, ..."
optional
  connection = "connection ID">
deleteAttachments
<cfexchangetask
  required
  action = "deleteAttachments"
  uid = "task UID"
optional
  connection = "connection ID">

get
<cfexchangetask
  required
  action = "get"
  name = "query identifier"
optional
  connection = "connection ID">

getAttachments
<cfexchangetask
  required
  action = "getAttachments"
  name = "query identifier"
  uid = "task UID"
optional
  attachmentPath = "directory path"
  connection = "connection ID"
  generateUniqueFilenames = "no|yes”>

modify
<cfexchangetask
  required
  action = "modify"
  task = "#task information structure#"
  uid = "task UID"
optional
  connection = "connection ID"
Note

If you omit the connection attribute, create a temporary connection by specifying cfexchangecollection tag attributes in the cfexchangetask tag. In this case, ColdFusion closes the connection when the tag completes. For details, see the cfexchangeconnection tag open action.

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also
cfexchangecalendar, cfexchangeconnection, cfexchangecontact, cfexchangefilter, cfexchangemail, Interacting with Microsoft Exchange Servers in the Developing ColdFusion Applications.

Attributes

The following table provides detailed information about each attribute. It lists the attribute name, the actions (action attribute values) to which it applies, whether it is required or optional for those actions, and its default value, if any, and provides a detailed description of the attribute and its valid values.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>all</td>
<td>Required</td>
<td>Default</td>
<td>The action to take. Must be one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• create</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• deleteAttachments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• get</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• getAttachments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• modify</td>
</tr>
<tr>
<td>attribute</td>
<td>action</td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>attachmentPath</td>
<td>getAttachments</td>
<td>Optional</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The filepath of the directory in which to put the attachments. If the directory does not exist, ColdFusion creates it.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: If you omit this attribute, ColdFusion does not save any attachments. If you specify a relative path, the path root is the ColdFusion temporary directory, which is returned by the <code>GetTempDirectory</code> function.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>connection</td>
<td>all</td>
<td>Optional</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The name of the connection to the Exchange server, as specified in the <code>cfxchangeconnection</code> tag. If you omit this attribute, create a temporary connection by specifying <code>cfxchangeconnection</code> tag connection attributes in the <code>cfxchangetask</code> tag.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>generateUniqueFilenames</td>
<td>A Boolean value that specifies whether to generate unique filenames if multiple attachments have the same filenames. If two or more attachments have the same filename and this option is <code>yes</code>, ColdFusion appends a number to the filename body (before the extension) of any conflicting filenames. Thus, if three attachments have the name <code>myfile.txt</code>, ColdFusion saves the attachments as <code>myfile.txt</code>, <code>myfile1.txt</code>, and <code>myfile2.txt</code>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>The name of the ColdFusion query variable that contains the returned task records or information about the attachments that were retrieved. For more information on the returned data, see Usage.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>result</td>
<td>The name of a variable that contains the UID of the task that is created. You use this value in the <code>uid</code> attribute of other actions to identify the task to be acted on.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
</tbody>
</table>
| serverVersion | Optional | 2007  | Specifies the Microsoft Exchange Server version. The values are:  
- 2003  
- 2007  
- 2010  
If you do not specify the details, 2007 is taken by default. The value you specify overrides the value that you specify at the application level. |
| task       | Required |      | A reference to the structure that contains the task properties to be set or changed and their values. Specify this attribute in number signs (#). For more information on the event structure, see Usage. |
| uid        | Required |      | A case-sensitive Exchange UID value that uniquely identifies the tasks on which to perform the action. For the delete action, this attribute can be a comma-delimited list of UID values. The deleteAttachment, getAttachments, and modify actions allow only a single UID value. |

When you specify the create or modify action, the task attribute must specify a structure that contains information that defines the events. The structure can have the following fields. Include only the fields that you are setting or changing.
<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActualWork</td>
<td>A number in minutes. Cannot be less than zero.</td>
</tr>
<tr>
<td>Attachments</td>
<td>The pathnames of any attachments to include in the task. To specify multiple files, separate filepaths with semicolons (;) for Windows, and colons (:) for UNIX and Linux. Use absolute paths. If you specify one or more attachments for a modify action, these are added to any existing attachments; the pre-existing attachments are not deleted.</td>
</tr>
<tr>
<td>Categories</td>
<td>A comma-delimited list of categories. The filter searches for tasks that match all the categories in the list.</td>
</tr>
<tr>
<td>BillingInfo</td>
<td>A string.</td>
</tr>
<tr>
<td>Companies</td>
<td>A string.</td>
</tr>
<tr>
<td>DateCompleted</td>
<td>A string in a date format that is valid in ColdFusion. If you omit this field and set the Status field to completed, or set the PercentCompleted field to 100, this value is set to the current date. If you set this date, the Status value is set to Completed and the PercentCompleted field is set to 100.</td>
</tr>
<tr>
<td>DueDate</td>
<td>A string in a date format that is valid in ColdFusion.</td>
</tr>
<tr>
<td>Message</td>
<td>A string containing the task description.</td>
</tr>
<tr>
<td>Mileage</td>
<td>A string.</td>
</tr>
<tr>
<td>PercentCompleted</td>
<td>A number in the range 0-100. If you set this field to 100, The following values are set:</td>
</tr>
<tr>
<td></td>
<td>• The Status value is set to Completed.</td>
</tr>
<tr>
<td></td>
<td>• If the DateCompleted value is or was not set, it is set to the current date. If you set this value to a number with a value less than 100, the following values are set:</td>
</tr>
<tr>
<td></td>
<td>• If Status field is or was set to Completed, the Status is set to In_Progress.</td>
</tr>
<tr>
<td></td>
<td>• The DateCompleted value is cleared.</td>
</tr>
<tr>
<td>Priority</td>
<td>One of the following values:</td>
</tr>
<tr>
<td></td>
<td>• low</td>
</tr>
<tr>
<td></td>
<td>• normal</td>
</tr>
<tr>
<td></td>
<td>• high</td>
</tr>
<tr>
<td><strong>ReminderDate</strong></td>
<td>A string in a date format that is valid in ColdFusion.</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>StartDate</strong></td>
<td>A string in a date format that is valid in ColdFusion. When you create a task, the default value defaults is the current date.</td>
</tr>
</tbody>
</table>
| **Status**       | The following values are valid: Not Started, In Progress, Completed, Waiting, or Deferred. If you omit this field and the PercentCompleted value is less than 100, the Status value is set to In Progress. If you set this field to Completed, the following values are also set:  
  - The PercentCompleted value is set to 100.  
  - If the DateCompleted value is not set, it is set to the current date.  
  If you set this field to a value other than Completed, the following values are also set:  
  - If the PercentCompleted field is or was 100, the PercentCompleted value is reset to 0.  
  - The DateCompleted value is set to 0. |
| **Subject**      | A String. |
| **TotalWork**    | A number in minutes. Cannot be less than zero. |

**Usage**

The `cfexchangetask` tag manages task records on the Exchange server. Use the `cfexchangetask` tag to perform the following actions:

- Create a task.
- Delete one or more tasks.
- Get one or more task records that conform to an optional set of filter specifications, such as the last name, job title, or home phone number, and so on.
- Get the attachments for a specific task record.
- Modify an existing task

To use this tag, you must have a connection to an Exchange server. If you are using multiple tags that interact with the exchange server, such as if you are creating several task records, use the `cfexchangeconnection` tag to create a persistent connection. You then specify the connection identifier in each `cfexchangetask`, or any other ColdFusion Exchange tag, if you are also accessing calendar entries, contacts, or mail. Doing this saves the overhead of creating and closing the connection for each tag. Alternatively, you can create a temporary connection that lasts only for the time that ColdFusion processes the single `cfexchangetask` tag. To do this, you specify the connection attributes directly in the `cfexchangetask` tag. For details on the connection attributes, see the `cfexchangeconnection` tag.

**The delete action**

When you specify the delete action, specify a `uid` attribute with a comma-delimited list of one or more Exchange UIDs that identify the tasks to delete. You can use the get action, with an appropriate filter expression, to determine
the UID values to specify. If all UIDs that you specify are invalid, the `cfexchangetask` tag generates an error. If at least one UID is valid, the tag ignores any invalid UIDs and deletes the items specified by the valid UID.

**The get action**

When you specify the `get` action, the query object specified by the `name` attribute contains one record for each retrieved task. The query object has columns with the same names and data formats as the fields listed for the `task` attribute structure, with the following differences:

- The query object has a Boolean `HasAttachment` column, and does not have an `Attachments` column. If the `HasAttachment` field value is `yes`, use the `getAttachments` action to retrieve the attachments.
- The query object has an additional UID column with the unique identifier for the task in the Exchange server. You can use this value in the `uid` attribute of the `getAttachments`, `delete`, and `modify` actions to identify the required task.
- The query object has an additional `HtmlMessage` column. The `Message` column has a plain-text version of the task description, and the `HtmlMessage` column text includes the description's HTML formatting. You use child `cfexchange` tags to specify the messages to get. For detailed information, see under `cfexchangefilter`.

**The getAttachments action**

When you use the `getAttachments` action, specify a single UID and a `name` attribute. The `cfexchangetask` tag populates a query object specified by the `name` attribute with the specified name. Each record has the following information about an attachment to the specified task:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachmentFileName</td>
<td>The filename of the attachment.</td>
</tr>
<tr>
<td>attachmentFilePath</td>
<td>The absolute path of the attachment file on the server. If you omit the <code>attachmentPath</code> attribute, this column contains the empty string.</td>
</tr>
<tr>
<td>CID</td>
<td>The content-ID of the attachment. Typically used in HTML img tags to embed images in a message.</td>
</tr>
<tr>
<td>mimeType</td>
<td>The MIME type of the attachment, such as text/html</td>
</tr>
<tr>
<td>isMessage</td>
<td>A Boolean value that specifies whether the attachment is a message.</td>
</tr>
<tr>
<td>size</td>
<td>The attachment size, in bytes.</td>
</tr>
</tbody>
</table>

The tag places the attachments in the directory specified by the `attachmentPath` attribute. If you omit the `attachmentPath` attribute, ColdFusion does not get any attachments, it gets the information about the attachments. This lets you determine the attachments without incurring the overhead of getting the attachment files.Use the following syntax to specify an in-memory `attachmentPath` directory. In-memory files are not written to disk and speed processing of transient data.
attachmentpath = "ram:///path"

The path can include multiple directories, for example ram:///petStore/orders/messageAttachments. Create all directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications. The getAttachments action works only if authentication for EWS (Exchange Web Services) is set to basic in the server setup of Exchange. IWA (Integrated Windows Authentication) is not supported.

The modify action

If you specify the modify action, the uid attribute must specify a single Exchange UID. The task structure must specify only the fields that you are changing. Any fields that you do not specify remain unchanged. For a detailed description of the contents of the task structure, see the Attributes section. If a task has attachments and you specify attachments when you modify the task, the new attachments are added to the previous attachments, and do not replace them. Use the deleteAttachments action to remove any attachments.

Example

The following example uses a transient connection to create a single task:

```cfscript>
stask=StructNew();
stask.Priority="high";
stask.Status="Not_Started";
stask.DueDate="3:00 PM 09/14/2007";
stask.Subject="My New Task";
stask.PercentCompleted=0;
Message="Do this NOW!";
</cfscript>

<!--- Create the task using a transient connection. --->
<cfexchangetask action="create"
username="#user1#"
password="#password1#"
server="#exchangeServerIP#"
task="#stask#"
result="theUID">

<!--- display the UID to confirm that the action completed. --->
<cfdump var="#theUID#"/>
**cfexecute**

**Description**

Executes a ColdFusion developer-specified process on a server computer.

**Category**

Extensibility tags, Flow-control tags

**Syntax**

```xml
<cfexecute
    name = "application name"
    arguments = "command line arguments"
    outputFile = "output filename"
    errorFile = "filename to store error output"
    timeout = "timeout interval"
    variable = "variable name"
    errorVariable = "variable name">
    ...
</cfexecute>
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfcollection, cfindex, cfobject, cfreport, cfsearch, cfwddx

**History**

ColdFusion MX 6.1:

- Added the variable attribute.
- Changed filepath behavior for the outputFile attribute: if you do not specify an absolute filepath in the outputFile attribute, the path is relative to the ColdFusion temporary directory.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Absolute path of the application to execute. On Windows, specify an extension, for example, C:\myapp.exe.</td>
</tr>
</tbody>
</table>
| **arguments** | Optional | Command-line variables passed to application. If specified as string, it is processed as follows:

- Windows: passed to process control subsystem for parsing.
- UNIX: tokenized into an array of arguments. The default token separator is a space; you can delimit arguments that have embedded spaces with double-quotations marks.

If passed as array, it is processed as follows:

- Windows: elements are concatenated into a string of tokens, separated by spaces. Passed to process control subsystem for parsing.
- UNIX: elements are copied into an array of exec() arguments. |

| **outputFile** | Optional | File to which to direct program output. If no `outputFile` variable attribute is specified, output is displayed on the page from which it was called. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function. |

| **errorFile** | Optional | File in which to direct error output, if any exists. |
| **timeout** | Optional | 0 | Length of time, in seconds, that ColdFusion waits for output from the spawned program.  
- 0: equivalent to nonblocking mode.  
- A very high value: equivalent to blocking mode.  
  If the value is 0:  
  - ColdFusion starts a process and returns immediately. ColdFusion may return control to the calling page before any program output displays. To ensure that program output displays, set the value to 2 or higher.  
  - If the **outputFile** attribute is not specified, any program output is discarded |
| **variable** | Optional | | Variable in which to put program output. If no **outputFile** or **variable** attribute is specified, output is displayed on page from which it was called |
| **errorVariable** | Optional | | Variable in which to put error output, if any exists |

**Usage**

Do not put other ColdFusion tags or functions between the start and end tags of `cfexecute`. You cannot nest `cfexecute` tags.

**Exceptions**

Throws the following exceptions:

- If the application name is not found: `java.io.IOException`
- If the effective user of the ColdFusion executing thread does not have permissions to execute the process: a security exception  
The time-out values must be between zero and the longest time-out value supported by the operating system.

**Example**
<h3>cfexecute</h3>

This example executes the Windows NT version of the netstat network monitoring program, and places its output in a file.

```cfexecute
  name = "C:\WinNT\System32\netstat.exe"
  arguments = "-e"
  outputFile = "C:\Temp\output.txt"
  timeout = "1"
</cfexecute>
### cfexit

**Description**

This tag aborts processing of the currently executing CFML custom tag, exits the page within the currently executing CFML custom tag, or re-executes a section of code within the currently executing CFML custom tag.

**Category**

*Debugging tags, Flow-control tags*

**Syntax**

```xml
<cfexit
    method = "method"/>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

`cfabort`, `cfbreak`, `cfexecute`, `cfif`, `cflocation`, `cfloop`, `cfswitch`, `cfthrow`, `cftry`, `cfabort` and `cfexit` in the *Developing ColdFusion Applications*

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| method    | Optional| exitTag | • `exitTag`: aborts processing of currently executing tag.  
• `exitTemplate`: exits page of currently executing tag.  
• `loop`: re-executes body of currently executing tag. |

**Usage**

If this tag is encountered outside the context of a custom tag, for example in the base page or an included page, it executes in the same way as `cfabort`. The `cfexit` tag can help simplify error checking and validation logic in custom tags. The `cfexit` tag function depends on its location and execution mode:

<table>
<thead>
<tr>
<th>Method value</th>
<th>Location of cfexit call</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>exitTag</code></td>
<td>Base page</td>
<td>Terminate processing</td>
</tr>
<tr>
<td>Command</td>
<td>Execution mode</td>
<td>Action</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>exitTemplate</td>
<td>Execution mode = Start</td>
<td>Continue after end tag</td>
</tr>
<tr>
<td></td>
<td>Execution mode = End</td>
<td>Continue after end tag</td>
</tr>
<tr>
<td>loop</td>
<td>Base page</td>
<td>Terminate processing</td>
</tr>
<tr>
<td></td>
<td>Execution mode = Start</td>
<td>Continue from first child in body</td>
</tr>
<tr>
<td></td>
<td>Execution mode = End</td>
<td>Continue after end tag</td>
</tr>
<tr>
<td></td>
<td>Execution mode = Start</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td>Execution mode = End</td>
<td>Continue from first child in body</td>
</tr>
</tbody>
</table>

**Example**
<h3>cfexit Example</h3>
<p>cfexit can be used to abort the processing of the currently executing CFML custom tag. Execution resumes following the invocation of the custom tag in the page that called the tag.</p>

<h3>Usage of cfexit</h3>
<p>cfexit is used primarily to perform a conditional stop of processing inside a custom tag. cfexit returns control to the page that called that custom tag, or in the case of a tag called by another tag, to the calling tag.</p>

<!---- cfexit can be used within a CFML custom tag, as follows: --->
<!---- Place this code (uncomment the appropriate sections) within the customtags directory. --->

<!---- MyCustomTag.cfm --->
<!---- This simple custom tag checks for the existence of myValue1 and myValue2. If they are both defined, the tag adds them and returns the result to the calling page in the variable "result". If either or both of the expected attribute variables is not present, an error message is generated, and cfexit returns control to the calling page. --->

<!---- <cfif NOT IsDefined("attributes.myValue2")> -->
<!---- <cfset caller.result = "Value2 is not defined"> -->
<!---- <cfexit method = "exitTag"> -->
<!---- <cfelseif NOT IsDefined("attributes.myValue1")> -->
<!---- <cfset caller.result = "Value1 is not defined"> -->
<!---- <cfexit method = "exitTag"> -->
<!---- <cfelse> -->
<!---- <cfset value1 = attributes.myValue1> -->
<!---- <cfset value2 = attributes.myValue2> -->
<!---- <cfset caller.result = value1 + value2> -->
<!---- </cfif> --->
<!---- End MyCustomTag.cfm --->

<!---- Place this code within your page --->

<!---- <p>The call to the custom tag, and then the result: </p> -->
<!---- <CF_myCustomTag myvalue2 = 4> -->
<!---- <cfoutput>#result#</cfoutput> --->
<!---- If cfexit is used outside a custom tag, it functions like a cfabort. For example, the text after this message is not processed: </p> -->
<!---- cfexit> -->
<!---- This text is not executed because of the cfexit tag above it. </p>
Tags f

This section lists tags starting with alphabet F with their description.

- cffield
- cfile
  - cfile action = "append"
  - cfile action = "copy"
  - cfile action = "delete"
  - cfile action = "move"
  - cfile action = "read"
  - cfile action = "readBinary"
  - cfile action = "rename"
  - cfile action = "upload"
  - cfile action = "uploadAll"
  - cfile action = "write"
- cfileupload
- cfinally
- cfflush
- cfformgroup
- cfformitem
- cfftp
  - cfftp: Opening and closing FTP server connections
  - cfftp: Opening and closing secure FTP server connections
  - cfftp: Connection: file and directory operations
  - cfftp action = "listDir"
- cfform
- cffunction
cfeed

Description

Reads or creates an RSS or Atom syndication feed. This tag can read RSS versions 0.90, 0.91, 0.92, 0.93, 0.94, 1.0, and 2.0, and Atom 0.3 or 1.0. It can create RSS 2.0 or Atom 1.0 feeds.

Category

Communications tags, Internet protocol tags

Syntax
create
required
<cffeed
action = "create"
name = "#structure#"
One or both of the following:  outputFile = "path"
xmlVar = "variable name"
optional
overwrite = "no|yes">
escapeChars = "true|false">

OR

required
<cffeed
action = "create"
properties = "#metadata structure#"
query = "#items/entries query name#"
One or both of the following:  outputFile = "path"
xmlVar = "variable name"
optional
columnMap = "mapping structure"
overwrite = "no|yes">

read
required
<cffeed
source = "feed source"
One or more of the following:  name = "structure"
properties = "metadata structure"
query = "items/entries query"
outputFile = "path"
xmlVar = "variable name"
optional
action = "read"
enclosureDir = "path"
ignoreEnclosureError = "no|yes"
overwrite = "no|yes"
overwriteEnclosure = "no|yes"
proxyServer = "IP address or server name for proxy host"
proxyPassword = "password for the proxy host"
proxyPort = "port of the proxy host"
proxyUser = "user name for the proxy host"
timeout = "request time-out in seconds"
userAgent = "HTTP user agent identifier">

⚠️ Note
You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.
ColdFusion 8: Added this tag.

ColdFusion 9: The escapeChars attribute is newly added.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| action    | Optional| read    | The action to take, one of the following values:  
  - create: creates an RSS 2.0 or Atom 1.0 feed XML document and saves it in a variable, writes it to a file, or both.  
  - read: parses an RSS or Atom feed from a URL or an XML file and saves it in a structure or query. You can also get feed metadata in a separate structure. |
<p>| columnMap | Optional|         | Used only for the create action with a query attribute. A structure that specifies a mapping between the names of the columns in the object specified by the query attribute and the columns of the ColdFusion feed format (see the section Query object rules. The key for each field must be a column name (see the table in the Query object rules section). The value of the field must be the name of the corresponding column in the query object used as input to the create action. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enclosureDir</td>
<td>Optional</td>
<td></td>
<td>Used only for the <code>read</code> action. Path to the directory in which to save any enclosures that are available in the feed being read. The path can be absolute or relative to the CFML file. If the directory does not exist, ColdFusion generates an error. If you omit this attribute, ColdFusion does not save enclosures. To specify the directory that contains the current page, set this attribute to &quot;.&quot; (period).</td>
</tr>
<tr>
<td>escapeChars</td>
<td>Optional</td>
<td>false</td>
<td>Used only for the <code>create</code> action. If this attribute is <code>true</code>, escapes/replaces all the invalid characters as per W3C specification. <strong>Note</strong>: Japanese characters that are not part of UTF-8 encoding are replaced. Non-UTF-8 Japanese characters remain in the feed as they are. If this attribute is <code>false</code>, does not escape invalid characters and tries to generate feed. If JDOM fails to write the file with these invalid characters, the error message &quot;Invalid Character in Input&quot; is displayed.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ignoreEnclosureError</td>
<td>Optional</td>
<td>no</td>
<td>If this attribute is yes, ColdFusion attempts to save all enclosures. If it encounters an error downloading one enclosure, it continues downloading other enclosures and writes the error information in the server log. If this attribute is no, when ColdFusion encounters an error downloading an enclosure, it stops downloading all enclosures and generates an error. <strong>Note:</strong> Enclosure errors can occur if the specified enclosure is of a type that the web server does not allow to be downloaded.</td>
</tr>
</tbody>
</table>
| name               | See Note |         | A structure that contains complete feed data:  
  - The output of a read action.  
  - The input definition of the feed to create.  
When you specify the name attribute for a create action, enclose it in number signs (#).  
For more information, see Name and properties structure rules section. |
| outputFile         | See Note |         | Path of the file in which to write the feed as XML text.  
The path can be absolute, or relative to the CFML file. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>overwrite</td>
<td>Optional</td>
<td>no</td>
<td>Whether to overwrite the XML feed file if it exists. If you do not set this attribute to yes and the cffeed tag tries to write to a file that exists, ColdFusion generates an error.</td>
</tr>
<tr>
<td>overwriteEnclosure</td>
<td>Optional</td>
<td>no</td>
<td>Used only for the read action. Whether to overwrite files in the enclosure directory if they exist. If you do not set this attribute to yes and the cffeed tag tries to write to a file that exists, ColdFusion generates an error.</td>
</tr>
</tbody>
</table>
| properties         | See Note|       | A structure that contains the feed metadata, the information about the entire feed. Can contain either of the following:  
- The output of a read action.  
- Input to a create action.  
The properties and query attributes combined provide complete feed information. When you specify the properties attribute for a create action, enclose it in number signs (#). For more information, see Name and properties structure rules section. |
<p>| proxyPassword      | Optional|       | Password required by the proxy server.                                      |
| proxyPort          | Optional| 80    | The port to connect to on the proxy server.                                  |</p>
<table>
<thead>
<tr>
<th>proxyServer</th>
<th>Optional</th>
<th>Host name or IP address of a proxy server to which to send the request.</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxyUser</td>
<td>Optional</td>
<td>User name to provide to the proxy server.</td>
</tr>
<tr>
<td>query</td>
<td>See Note</td>
<td>A query object that contains the Atom entries or RSS items in the feed. Can contain either of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The output of a read action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Input to a create action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The properties and query attributes combined provide complete feed information. When you specify the query attribute for a create action, enclose it in number signs (#). For more information, see section Query object rules.</td>
</tr>
<tr>
<td>source</td>
<td>Required</td>
<td>Used only for the read action. The URL of the feed or the path to the XML file that contains the feed contents. A path can be absolute, or relative to the CFML file.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional</td>
<td>Request time-out</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The number of seconds to wait for a response from the feed source. A value of 0 specifies that the request does not time out. By default, ColdFusion uses the request time-out setting of the ColdFusion Administrator Server Settings &gt; Settings page.</td>
</tr>
<tr>
<td>userAgent</td>
<td>Optional</td>
<td>Cold Fusion</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>xmlVar</td>
<td>See Note</td>
<td></td>
</tr>
</tbody>
</table>

Usage

Specifying file and directory attributes

Use the following syntax to specify an in-memory file or directory. In-memory files are not written to disk and speed processing of transient data.

```
ram:///filepath
```

A filepath can include multiple directories, for example `ram:///petStore/images/poodle.jpg`. Create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications.

Setting and getting feed information

The `cffeed` tag lets you specify and save feed data in many, flexible ways.

Usage

Use the following syntax to specify an in-memory file, which is not written to disk. In-memory files speed processing of transient data.

```
ram:///filepath
```

The filepath can include directories, for example `ram:///petStore/images/poodle.jpg`. Create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications.

When you create a feed

- You specify the feed data in either of the following ways:
  - By putting all metadata and entry or item data in a single structure specified by the `name` attribute.
  - By putting the metadata in a structure specified by the `properties` structure and the entries or items as rows in a query object specified by the `query` attribute.
- You save the resulting feed XML in one or both of the following places:
  - A file specified by the `OutputFile` attribute. The `cffeed` tag saves the data in UTF-8 encoding.
  - An variable specified by the `xmlVar` attribute

When you read a feed You can save the feed data in any combination of the following forms:
By saving all entry or item data and metadata in a single structure specified by the `name` attribute
By saving entries or items as rows in a query object specified by the `query` attribute
By saving the metadata in a structure specified by the `properties` structure
By writing the feed XML in a file specified by the `OutputFile` attribute. The `cffeed` tag saves the data in UTF-8 encoding.
By saving the feed XML in a ColdFusion XML variable specified by the `xmlVar` attribute

When you save feed data, you do not have to save both the metadata and the entry or item data. You can specify only the `properties` attribute, or only the `query` attribute.

### Name and properties structure rules

The `name` and `properties` structures must conform to the following rules. For more information on requirements for specific metadata entries, see the section Representing feed metadata.

- All structure key names must be identical to the corresponding feed element names, with the exception of the `version` and `encoding` fields. Also, the key names for Dublin Core and Apple iTunes extension elements start with `DC_` and `ITUNES_` respectively.
- The `properties` structure fields are identical to the metadata fields in the `name` structure.
- When you read a feed, the structure contains only those elements and attribute values that exist in the feed. For requirements for the `create` action, see the section Creating feeds.
- If the feed can have multiple elements of the same type (such as `entry`, `item`, or `link`), the `name` or `property` structure has a single entry that contains the data for all of the elements. The structure entry has the following format:
  - The key is the element name (for example, `item`)
  - The value is an array of structures
  - Each structure in the array represents one element.

ColdFusion uses an array even if there is only a single element. If an Atom feed has only one `link` element, for example, you must specify that element in a `name` attribute structure by using the following format:

```
structureName.link[1]
```

For example, to specify a `link` metadata entry in an Atom 1.0 feed, you could use the following code:

```
<cfset meta.link = arrayNew(1)>
<cfset meta.link[1] = structNew()>
<cfset meta.link[1].href = "http://www.myCo,com">
```

- If an element can have multiple attributes, or can have at least one attribute and a value, the element is represented as a structure, even if the element specifies only one attribute or only a value.
- If an element has one or more attributes and a value (body), the value is in a field of the element structure named `value`. For example, the text of the `summary` element for the third `entry` in an Atom feed would go in a field whose name has the following format:

  ```
  structureName.entry[3].summary.value.
  ```

- When the `cffeed` tag reads a feed, it reports dates as follows: Atom: W3C date format, such as
Query object rules

The query object specified by the *query* attribute conforms to the following rules:

- The query object format supports multiple feed formats, and many feeds do not include all optional feed attributes or elements. As a result:
  - When you read a feed, the returned query object contains entries for all standard RSS and Atom fields, even for fields that are not supported by the feed type. Any columns that are not used by the feed format, or are not used in that specific feed, contain empty strings or undefined values.
  - When you read a feed, the query object contains all iTunes extension fields if the feed contains any iTunes extension elements, and the query object contains all Dublin Core extension fields if the feed contains any Dublin Core extension elements. Otherwise, the query results do not contain any of the extension fields.
  - When you create a feed, the query that you define requires only those columns that contain data for your feed; you can omit unused columns.
- If a feed entry or item has multiple child elements with the same name, the query column represents the element values as a comma-delimited list. RSS 2.0 items can have multiple *category* elements. Atom 1.0 entries can have multiple *category*, *author*, *contributor*, and *link* elements. The Dublin Core extensions allow all multiples of all element types.
- Many entry or item elements that can have multiple instances have multiple attributes, not all of which are required for any particular element instance. If an entry or item has multiple instances of an element, and any of those elements omit attributes, ColdFusion represents the omitted attribute in the lists by a space. In XML, an Atom entry, for example, might contain three author elements, as follows:

  ```xml
  <author>
    <person>Anthony</person>
    <uri>http://www.MyCo.com</uri>
    <email>Tony@MyCo.com</email>
  </author>
  <author>
    <person>Beverly</person>
  </author>
  <author>
    <person>Cathy</person>
    <email>cathy@MyCo.com</email>
  </author>
  
  The ColdFusion query represents these columns as follows:

<table>
<thead>
<tr>
<th>AUTHOR_PERSON</th>
<th>AUTHOR_URI</th>
<th>AUTHOR_EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anthony, Beverly, Cathy</td>
<td><a href="http://www.MyCo.com">http://www.MyCo.com</a>,</td>
<td><a href="mailto:Tony@MyCo.com">Tony@MyCo.com</a>, ,<a href="mailto:cathy@MyCo.com">cathy@MyCo.com</a></td>
</tr>
</tbody>
</table>

The following table lists the columns of the standard query object specified by the *query* attribute. If an RSS feed includes either Dublin Core extensions or iTunes extensions, the query includes additional columns. For information on these fields, see the sections Dublin Core Extensions and Apple iTunes Extensions.
<table>
<thead>
<tr>
<th>Column</th>
<th>Atom entry</th>
<th>RSS item</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHOREMAIL</td>
<td>author element email attribute</td>
<td>author item</td>
</tr>
<tr>
<td>AUTHORNAME</td>
<td>author element name attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>AUTHORURI</td>
<td>author element uri attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>CATEGORYLABEL</td>
<td>category element label attribute</td>
<td>category item value</td>
</tr>
<tr>
<td>CATEGORYSCHEME</td>
<td>category element scheme attribute</td>
<td>category item domain attribute</td>
</tr>
<tr>
<td>CATEGORYTERM</td>
<td>category element term attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>Not used</td>
<td>comments item value</td>
</tr>
<tr>
<td>CONTENT</td>
<td>content element value</td>
<td>description item value</td>
</tr>
<tr>
<td>CONTENTMODE</td>
<td>content element mode attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>CONTENTSRC</td>
<td>content element src attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>CONTENTTYPE</td>
<td>content element type attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>CONTRIBUTOREMAIL</td>
<td>contributor element email attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>CONTRIBUTORNAME</td>
<td>contributor element name attrib ute</td>
<td>Not used</td>
</tr>
<tr>
<td>CONTRIBUTORURI</td>
<td>contributor element uri attrib ute</td>
<td>Not used</td>
</tr>
<tr>
<td>CREATEDDATE</td>
<td>created element value (Atom 0.3 only)</td>
<td>Not used</td>
</tr>
<tr>
<td>EXPIRATIONDATE</td>
<td>Not used</td>
<td>expirationDate item value (RSS 0.93 only)</td>
</tr>
<tr>
<td>ID</td>
<td>id element value</td>
<td>guid item value</td>
</tr>
<tr>
<td>IDPERMALINK</td>
<td>Not used</td>
<td>guid item ispermalink attribute</td>
</tr>
<tr>
<td>LINKHREF</td>
<td>link element href attribute</td>
<td>enclosure item url attribute</td>
</tr>
<tr>
<td>LINKHREFLANG</td>
<td>link element hreflang attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>LINKLENGTH</td>
<td>link element length attribute</td>
<td>enclosure item length attribute</td>
</tr>
<tr>
<td>LINKREL</td>
<td>link element rel attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>LINKTITLE</td>
<td>link element title attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>LINKTYPE</td>
<td>link element type attribute</td>
<td>enclosure item type attribute</td>
</tr>
<tr>
<td>PUBLISHEDDATE</td>
<td>published element value (issued in Atom 0.3)</td>
<td>pubDate item value</td>
</tr>
<tr>
<td>RIGHTS</td>
<td>rights element value (copyright in Atom 0.3)</td>
<td>Not used</td>
</tr>
<tr>
<td>RSSLINK</td>
<td>Not used</td>
<td>link item value</td>
</tr>
<tr>
<td>SOURCE</td>
<td>Not used</td>
<td>source item value</td>
</tr>
<tr>
<td>SOURCEURL</td>
<td>Not used</td>
<td>source item url attribute</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>summary element value</td>
<td>Not used</td>
</tr>
<tr>
<td>SUMMARYMODE</td>
<td>summary element mode attribute</td>
<td>(Atom 0.3 only)</td>
</tr>
<tr>
<td>SUMMARYSRC</td>
<td>Blank for all well-formed Atom feeds. Contains data only if an Atom 1.0 feed uses a content element format for the summary element.</td>
<td>Not used</td>
</tr>
<tr>
<td>SUMMARYTYPE</td>
<td>summary element type attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>TITLE</td>
<td>title element value</td>
<td>title item value</td>
</tr>
<tr>
<td>TITLETYPE</td>
<td>title element type attribute</td>
<td>Not used</td>
</tr>
<tr>
<td>UPDATEDDATE</td>
<td>updated element value (modified in Atom 0.3)</td>
<td>Not used</td>
</tr>
<tr>
<td>URI</td>
<td>Not used</td>
<td>RSS 1.0 link item rdf:about attribute</td>
</tr>
<tr>
<td>XMLBASE</td>
<td>content element{{t xml:base}} attribute</td>
<td>Not used</td>
</tr>
</tbody>
</table>
Representing feed metadata

When you create a feed, the `name` and `properties` structures can represent all standard metadata for RSS 2 or Atom 1 feeds, in the format described in the Name and properties structure rules section. Similarly, when you read a feed, the structures represent all received metadata. The following rules apply to specific feed metadata fields in the `name` and `properties` structures:

- The `version` field identifies or specifies the feed version in the form `format_versionNumber`. For the `create` action, specify `atom_1.0` or `rss_2.0`. When you read an RSS 0.91 feed, the version field value is `rss_0.91`, not `rss_0.9`. For the `read` action, an `encoding` field identifies the XML encoding attribute, such as `iso-8859-1`. Do not specify an `encoding` field for a `create` action. Currently, ColdFusion generates all feeds in UTF-8 format and ignores any `encoding` value that you specify.
- The `feedExtension` field identifies whether the feed includes iTunes or Dublin Core extension content. Valid values are `itunes` and `DublinCore`. You do not have to specify this field when you create a feed with iTunes extensions; ColdFusion automatically determines that you have specified extension fields. (You cannot create a feed with Dublin Core extensions.)
- For the `read` action, an `encoding` field identifies the XML encoding attribute, such as `iso-8859-1`. Do not specify an `encoding` field for a `create` action. Currently, ColdFusion generates all feeds in UTF-8 format and ignores any `encoding` value that you specify.
- For RSS feeds, the `skihours` field contains a comma-delimited list of up to 24 numbers in the range 0-23, specifying hours of the day when aggregators should not read the feed. The hour beginning at midnight is hour zero. Your application can use the field to decide when to read the feed.
- For RSS feeds, the `skipdays` field contains a comma-delimited list of up to seven day-name values, specifying days of the week when aggregators must not read the feed. The valid names are `Monday`, `Tuesday`, `Wednesday`, `Thursday`, `Friday`, `Saturday` and `Sunday`. Your application can use the field to decide when to read the feed.

Dublin Core Extensions

Dublin Core extension elements provide additional metadata about the feed or an item. You can use the `cffeed` tag to read feeds that include elements that conform to the Dublin Core Metadata Element Set specification as metadata (channel elements) or as item elements. For detailed information Dublin Core extension elements, see the Dublin Core Metadata Element Set specification. At the time this topic was written, this specification was available at [http://dublincore.org/documents/dces/](http://dublincore.org/documents/dces/). ColdFusion support for Dublin Core extensions has the following limitations:

- You cannot create feeds containing these elements.
- You cannot get Dublin Core extension elements that are contained in a top-level (metadata) `image` element. ColdFusion ignores these elements.
- ColdFusion supports only the Dublin Core Metadata Element Set. It does not support the additional Dublin Core Metadata Initiative elements and element refinements.

When feed items include the Dublin Core extensions, the query specified by a `query` attribute includes all of the columns listed in the following table. If the feed does not include any Dublin Core extension elements, the query does not include the columns. With the exception of the `DC_SUBJECT_TAXONOMURI` and `DC_SUBJECT_VALUE` columns, each column name (without the `DC_` prefix) corresponds directly to a Dublin Core extension element name.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC_CONTRIBUTOR</td>
<td>The people or organizations responsible for contributing to the resource</td>
</tr>
<tr>
<td>DC_COVERAGE</td>
<td>The extent of the content in the resource</td>
</tr>
<tr>
<td>DC_CREATORTITLE</td>
<td>The person or organization responsible for creating this resource</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>DC_DATETITLE</td>
<td>A date or date and time associated with this resource</td>
</tr>
<tr>
<td>DC_DESCRIPTIONTITLE</td>
<td>A summary of the resource contents</td>
</tr>
<tr>
<td>DC_FORMATTITLE</td>
<td>The file format, physical medium, or dimensions of the resource</td>
</tr>
<tr>
<td>DC_IDENTIFIERTITLE</td>
<td>A string that can be used to unambiguously identify the resource</td>
</tr>
<tr>
<td>DC_LANGUAGETITLE</td>
<td>The language in which the resource is written</td>
</tr>
<tr>
<td>DC_PUBLISHERTITLE</td>
<td>The person or organization responsible for making the resource available.</td>
</tr>
<tr>
<td>DC_RELATIONTITLE</td>
<td>The identifier of a related resource, typically.</td>
</tr>
<tr>
<td>DC_RIGHTTITLE</td>
<td>Information about the property rights for the resource.</td>
</tr>
<tr>
<td>DC SOURCETITLE</td>
<td>A reference to the material from which this resource was derived.</td>
</tr>
<tr>
<td>DC_SUBJECT TAXONOMYURI</td>
<td>The taxonomy URI attribute of the Dublin Core subject element</td>
</tr>
<tr>
<td>DC_SUBJECT_VALUETITLE</td>
<td>The value of the Dublin Core subject element; a string that defines the topic of the resource</td>
</tr>
<tr>
<td>DC_TITLETITLE</td>
<td>A name to use for the resource</td>
</tr>
<tr>
<td>DC_TYPETITLE</td>
<td>The nature or genre of the resource</td>
</tr>
</tbody>
</table>

When you get data for a feed that includes Dublin Core elements as a structure, the element names are identical to the query column names listed above, with the exception of the representation of the Dublin Core subject element. The structure format represents the subject element as a dc_subject entry, which consists of an array of structures. The structures in the array have keys with the names value, for the element value, and taxonomyURI, for the taxonomyURI attribute.

**Apple iTunes Extensions**

You can use the `cffeed` tag to create or read feeds that contain elements defined in the Apple iTunes RSS podcast specification. For detailed information on iTunes extension format, see the Apple iTunes RSS specification. At the time this topic was written, this specification was available at [http://www.apple.com/itunes/store/podcaststechspecs.html](http://www.apple.com/itunes/store/podcaststechspecs.html).

You can create feeds with only a subset of the iTunes RSS extensions. When you read a feed, ColdFusion ignores all iTunes extension elements that are not in the supported subset. The following table lists the names of structure
entries or query column names for the supported elements. (These names consist of the `ITUNES_` prefix followed by
the iTunes extension element name.) The table also indicates which elements are used in the metadata, which are
used in the individual items, and which can be used in both:

<table>
<thead>
<tr>
<th>Element</th>
<th>Used in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITUNES_AUTHOR</td>
<td>Both</td>
<td>Artist name</td>
</tr>
<tr>
<td>ITUNES_BLOCK</td>
<td>Both</td>
<td>a value of yes requests to prevent the podcast or item (episode) from appearing. When ColdFusion reads a feed your application must determine this field's value and take any appropriate action.</td>
</tr>
<tr>
<td>ITUNES_DURATION</td>
<td>Item</td>
<td>The length of the item in second, or in HH:MM:SS format.</td>
</tr>
<tr>
<td>ITUNES_EXPLICIT</td>
<td>Both</td>
<td>A string indicating whether the item or items contain explicit material. Valid values are yes, no, and clean.</td>
</tr>
<tr>
<td>ITUNES_KEYWORDS</td>
<td>Both</td>
<td>A comma-delimited list of words or phrases used when searching in the iTunes music store.</td>
</tr>
<tr>
<td>ITUNES_SUBTITLE</td>
<td>Both</td>
<td>Short description text, usually only a few words.</td>
</tr>
<tr>
<td>ITUNES_SUMMARY</td>
<td>Both</td>
<td>A longer description (up to 4000 characters).</td>
</tr>
</tbody>
</table>

You can also use the following channel elements in the `name` or `properties` structures.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itunes_category</td>
<td>A structure that specifies the iTunes Music Store category. The structure has two fields:</td>
</tr>
<tr>
<td></td>
<td>* category</td>
</tr>
<tr>
<td></td>
<td>* subcategory</td>
</tr>
<tr>
<td></td>
<td>Notice that these element names do not have the <code>itunes_</code> prefix.</td>
</tr>
<tr>
<td>itunes_image</td>
<td>The URL of the artwork for the podcast.</td>
</tr>
</tbody>
</table>
Creating feeds

When you create a feed, you specify the feed contents in a name structure or in the combination of a query object and a properties structure. The cf feed tag generates the feed XML and saves it to the variable specified by the xmlVar attribute, the file specified by the outputFile attribute, or both. To create an RSS 2.0 feed you must specify the following metadata fields in a name structure or in a properties structure. All other RSS2.0 metadata fields, and all item fields, are optional.

- title
- link
- description
- version (must be "rss_2.0")

The cf feed tag does not enforce any rules on the Atom feed structure that it creates. You are responsible for ensuring that the feed is valid.

In most cases, a database table uses column names that differ from the column names you must use to create the feed. Therefore, use the columnmap attribute to map the input query column names to the required column names. The attribute is a structure whose keys are the column names required by the cf feed tag and whose values are the corresponding input query columns.

⚠️ **Note**

Always capitalize the input query column names irrespective of whether the database column names are capitalized or not.

The following example creates a feed using the cf artgallery data source's orders table. It maps the orders table ORDERDATE column to the query publisheddate column, the ADDRESS column to the content column, and so on. The sample code then displays the generated query XML to show the results.
<!---- Get the feed data as a query from the orders table. --->
<cfquery name="getOrders" datasource="cfartgallery">
   SELECT * FROM orders
</cfquery>

<!--- Map the orders column names to the feed query column names. --->
<cfset columnMapStruct = StructNew()>
<cfset columnMapStruct.publisheddate = "ORDERDATE">
<cfset columnMapStruct.content = "ADDRESS">
<cfset columnMapStruct.title = "CUSTOMERFIRSTNAME">
<cfset columnMapStruct.rsslink = "ORDERID">

<!--- Set the feed metadata. --->
<cfset meta.title = "Art Orders">
<cfset meta.link = "http://feedlink">
<cfset meta.description = "Orders at the art gallery">
<cfset meta.version = "rss_2.0">

<!--- Create the feed. --->
<cffeed action="create" query="#getOrders#" properties="#meta#" columnMap="#columnMapStruct#" xmlvar="rssXML">
<cfdump var="#XMLParse(rssXML)#"

---

Reading feeds

The cffeed tag does not validate the feeds that it reads. It can read invalid or loosely formatted feeds, but ignores some or all of the invalid content. For example, if you put more than one rights element in the Atom feed (which invalidates the feed), the cffeed tag ignores the elements after the first one, and doesn't generate an error. Dates and times in feeds that are being read must be in W3C or RFC 822 format. ColdFusion can also read iTunes extension dates in the format normally used by the iTunes music store.

Example

The following example creates an RSS feed. Enter fields for the feed title, link, and description elements. Also enter title, link, and description fields for one item. A second item is optional. The application saves the feed in a createRSSOutput.xml file in the feedTest subdirectory of the directory that contains the CFML page.

<!---- Generate the feed when the user submits a filled in form. --->
<cfif isDefined("Form.Submit")>
<cfscript>
   // Create the feed data structure and add the metadata.
   myStruct = StructNew();
   mystruct.link = form.link;
   myStruct.title = form.title;
   mystruct.description = form.description;
   mystruct.pubDate = Now();
   mystruct.version = "rss_2.0";
</cfscript>
/* Add the feed items. A more sophisticated application would use dynamic variables 
    and support varying numbers of items. */
myStruct.item = ArrayNew(1);
myStruct.item[1] = StructNew();
myStruct.item[1].description = StructNew();
myStruct.item[1].description.value = form.item1text;
myStruct.item[1].link = form.item1link;
myStruct.item[1].pubDate = Now();
myStruct.item[1].title = form.item1title;
myStruct.item[2] = StructNew();
myStruct.item[2].description = StructNew();
myStruct.item[2].description.value = form.item2text;
myStruct.item[2].link = form.item2link;
myStruct.item[2].pubDate = Now();
myStruct.item[2].title = form.item2title;

</cfscript>

<!--- Generate the feed and save it to a file and variable. --->
<cffeed action = "create" 
    name = "#myStruct#"
    outputFile = "feedTest/createRSSOutput.xml"
    overwrite = "yes"
    xmlVar = "myXML"> 
</cfif>

<!--- The user input form. --->
<cfform format="xml" preservedata="yes" style="width:500" height="700">
    <cfformitem type = "text"> Enter The Feed Metadata</cfformitem>
    <cfinput type = "text" label = "title" name = "title" 
        style = "width:435" required = "yes"> <br />
    <cfinput type = "text" label = "link" name = "link" 
        style = "width:435" required = "yes" validate = "url"> <br />
    <cftextarea name = "description" 
        style = "width:435; height:70" required = "yes" />

    <cfformitem type = "text"> Enter Item 1</cfformitem>
    <cfinput type="text" label="title" name="item1title" 
        style="width:435" required="yes" > <br />
    <cfinput type="text" label="link" name="item1link" 
        style="width:435" required="yes" validate="url"> <br />
    <cftextarea name = "item1text" 
        style = "width:435; height:70" required = "yes" /> <br />

    <cfformitem type = "text"> Enter Item 2</cfformitem>
    <cfinput type = "text" label = "title" name = "item2title" 
        style = "width:435"> <br />
    <cfinput type = "text" label = "link" name = "item2link" 
        style="width:435" required="yes" validate="url" > <br />
    <cftextarea name = "item2text" 
        style = "width:435; height:70" /> <br />
The following application is a simple feed reader that handles RSS and Atom feeds. It displays the feed title; for each item or entry, it shows the title as a link, and shows the published date and the item or entry contents. To use this example to read the feed created by the first application, enter the URL for the file the application created, for example, http://localhost:8500/cffeed/feedTest/createRSSOutput.xml.

```cfml
<cfinput type = "Submit" name = "submit" value = "Submit" />
</cfform>

<!--- Process the feed data if the user submitted the form --->
<cfif isDefined("Form.Submit")>
  <cffeed source = "#theURL#"
    properties = "myProps"
    query = "myQuery">
    <!--- Display the feed output. Use conditional logic for to handle different feed formats. --->
    <cfoutput>
      <h2>#myProps.title#</h2>
    </cfoutput>
    <cfoutput query = "myQuery">
      <cfif myProps.version IS "atom_1.0">
        <h3><a href = "#linkhref#">#title#</a></h3>
        <p><b>Published:</b> #DateFormat(publisheddate)#</p>
      </cfif>
      <cfelse>
        <h3><a href = "#rsslink#">#title#</a></h3>
        <p><b>Published:</b> #publisheddate#</p>
      </cfif>
    </cfoutput>
  </cffeed>
</cfif>

<!--- The form for specifying the feed URL or file --->
<cfform name = "SetFeed" preserveData = "yes">
  Enter Feed URL:
  <cfinput type = "text" size = "60" name = "theURL"><br>
  <cfinput type = "Submit" name = "submit" value = "Submit">
</cfform>
```
**cffile**

**Description**

Manages interactions with server files. The following sections describe the actions of the `cffile` tag:

- `cffile action = "append"
- `cffile action = "copy"
- `cffile action = "delete"
- `cffile action = "move"
- `cffile action = "read"
- `cffile action = "readBinary"
- `cffile action = "rename"
- `cffile action = "upload"
- `cffile action = "uploadAll"
- `cffile action = "write"

**Note**

To execute, this tag must be enabled in the ColdFusion Administrator. For more information, see Configuring and Administering ColdFusion.

If your ColdFusion applications run on a server used by multiple customers, consider the security of the files that could be uploaded or manipulated by cffile. For more information, see Configuring and Administering ColdFusion.

**Category**

File management tags

**Syntax**

The tag syntax depends on the `action` attribute value. See the following sections.

**See also**

`cfdirectory`

**History**

ColdFusion 10: Modifications to the attribute `accept`}

ColdFusion 9: `{uploadAll action

ColdFusion 8: Support for reading and writing `cfimages`

ColdFusion MX 7:

- Added the `result` attribute, which allows you to specify an alternate variable in which to receive result parameters. Used for `action = "upload"` action.
- Added the `fixnewline` attribute for `action = "append" and action = "write"` actions.

ColdFusion MX 6.1:

- Changed file path requirements: if you do not specify an absolute file path, the path is relative to the ColdFusion temporary directory, which is returned by the `GetTempDirectory` function.
- Changed behavior for `action="read"`: if the file starts with a byte order mark (BOM) ColdFusion uses it to
determine the character encoding.

- **Changed behavior for action="upload" nameConflict="MakeUnique"** ColdFusion now makes filenames unique by appending a incrementing number, 1 for the first file, 2 for the second and so on, to the name. In ColdFusion, filenames were made unique by appending an additional "1" for each file, as in 1, 11, 111, and so on.

**ColdFusion MX:**

- **Changed use of slashes in paths:** you can use forward or backward slashes in paths on both UNIX and Windows systems.
- **Changed file hierarchy requirements:** ColdFusion does not require that you put files and directories that you manipulate with this tag below the root of the web server document directory.
- **Changed directory path requirements for the destination attribute:** a directory path that you specify in the destination attribute does not require a trailing slash.
- **Deprecated the system value of the attributes attribute.**
- **Deprecated the temporary value of the attributes attribute.** In ColdFusion, it is a synonym for normal. It might not work in later releases.
- **Changed the action attribute options read, write, append and move:** they support a new attribute, chars et.
- **The archive value of the attributes attribute is obsolete and has no effect.**

### Provide file content in the tag body

For cffile action = "append" and cffile action = "write", you can provide file content in the tag body. If you provide file content in both the tag body and the output attribute, it results in an error. In the following example, the text provided in the body is written to myfile.txt in the current directory. Now assume that the file does not exist, then a new file myfile.txt is created. If the file exists, it is overwritten.

```coldfusion
<!---- In this case, file content is passed via both - output attribute and inside tag body. Tag body will get preference --->
<cfset filename = expandpath('./myfile.txt')>
<cftry>
  <cffile action="write" file="#filename#">
    some tag body
  </cffile>
  <cfset content = FileRead(filename)>
  <cfoutput>File Length = #Len(content)#</cfoutput>
  <cfcatch type="any">  
    <cfoutput>
      #cfcatch.message#
    </cfoutput>
    <br>
  </cfcatch>
</cftry>
```

Similarly, if you use cffile action="append", the tag body content is appended to the contents of the file myfile.txt. To create an empty file, you have to provide at least a blank line in the tag body as shown in the following code:
Modifications to the attribute accept in ColdFusion 10

The attribute accept takes comma-separated list of any or all extensions, MIME type, or list of MIME types as values. Specify the extensions with a . prefix. That is, only .txt is supported and not txt, *.txt, or *.txt.

However, you can use * as a wildcard to accept all files.

- **If you specify filename extension as value**, the file is checked, only to ensure if it matches the list of extensions you specified in comma-separated list. If it matches, the file is uploaded.
- **If the value is MIME type or list of MIME types** and the attribute strict is set to true, then the first few bytes of the file are read to determine the MIME type. If MIME type matches with what you have specified, upload occurs, else results in an error. The default value of strict is false.
- **If you specify both filename extensions and MIME types** and if strict is set to false, the verification is based on the order in which you have specified the values. For example, if the first value is .txt, then the .txt file is uploaded.
  
  If strict is set to true, extensions are ignored.

Example: Using the attribute accept to verify the filename extension

upload.cfm

```coldfusion
<cftry>
  <cfset variables.URL = 
  "http://#cgi.server_name#:#cgi.server_port##getDirectoryFromPath(cgi.script_name)#/_upload.cfm">
  <cfhttp method="Post" url="#variables.URL#">
    <cfhttpparam type="FILE" name="myfile" file="#expandpath('./sample.txt')#">
  </cfhttp>
  <cfoutput>#{cfhttp.filecontent}#</cfoutput>
</cfcatch>
<cfcatch>
  <cfoutput>
    #CFCATCH.message#
  </cfcatch>
</cftry>
```

_upload.cfm
In this example, `accept` is set to `.txt` and `strict` is not specified (and therefore is `false` by default). So, only files with `.txt` extensions are considered for upload. Even if the file is originally a PDF (sample.pdf) renamed as `sample.txt`, the file is uploaded (since `strict` is not set to `true`).

If you specify `strict=true`, then if the file is originally a `.txt` (and not renamed from some other type) the file is uploaded only if the correct MIME type is specified. That is, `strict=true` requires MIME type to be specified in the `accept` attribute. So, you should explicitly say `accept="text/plain"` in _upload.cfm.

**Example 2: Using MIME type**

Modify the Example 1 by specifying `accept="application/pdf"` in _upload.cfm as follows:

```cfml
<cffile action="UPLOAD" 
  destination="#uploadDirectory#" 
  filefield="form.myFile" 
  nameconflict="MAKEUNIQUE " 
  accept="application/pdf " 
  strict=true>
```
Since `strict = true`, only files of type PDF are considered for upload. To use a different file, modify the following section of `upload.cfm`:

```cfhttpparam type="FILE" name="myfile" file="#expandpath('./sample.txt')#"```

**Example 3: Using both extension and MIME type**

Modify the `Example 1` by specifying the following in `_upload.cfm`:

```cffile
action="UPLOAD"
destination="#uploadDirectory#"
filefield="form.myFile"
namemulti="MAKEUNIQUE"
accept=".txt, application/pdf"
strict=true>
```

Only PDF files are uploaded because `strict = true`. If you set `strict` as `false`, then both the files are uploaded. To use a different file, modify the following snippet of `upload.cfm`:

```cfhttpparam type="FILE" name="myfile" file="#expandpath('./sample.txt')#"```

**Example**

```cif IsDefined("form.formsubmit") is "Yes">
  <!-- The form has been submitted, now do the action. -->
  <cffile action="new">
    <!-- Make a new file. -->
    <cffile action="Write">
      file="#GetTempDirectory()#foobar.txt"
      output="#form.the_text#"
    </cffile>
  </cffile>
  <cffile action="read">
    <!-- Read existing file. -->
    <cffile action="Read">
      file="#GetTempDirectory()#foobar.txt"
      variable="readText"
    </cffile>
  </cffile>
  <cffile action="add">
    <!-- Update existing file. -->
    <cffile action="Append">
      file="#GetTempDirectory()#foobar.txt"
      output="#form.the_text#"
    </cffile>
  </cffile>
```
<cfif form.action is "delete">
<!---- Delete existing file. ---->
<cfif action="Delete">
  file="#GetTempDirectory()#foobar.txt">
</cfif>
</cfif>
<!---- Set some variables. ---->
<cfparam name="fileExists" default="no">
<cfparam name="readText" default=""/>
<!---- First, check whether canned file exists. ---->
<cfif FileExists("#GetTempDirectory()#foobar.txt") is "Yes">
  <cfset fileExists="yes">
</cfif>
<!---- Now, make the form that runs the example. ---->
<form action="index.cfm" method="POST">
<h4>Type in some text to include in your file: </h4> <p>
<cfif fileExists is "yes">
<p>A file exists (foobar.txt, in <cfoutput>#GetTempDirectory()#</cfoutput>). </p>
You may add to it, read from it, or delete it. </p>
</cfif>
<!---- If reading from a form, let that information display in textarea. ---->
<textarea name="the_text" cols="40" rows="5">
<cfif readText is not "">
<cfoutput>#readText#</cfoutput>
</cfinput></textarea>
<!---- Select from the actions depending on whether the file exists. ---->
<select name="action">
<cfif fileExists is "no">
<option value="new">Make new file</option>
</cfif>
<cfif fileExists is "yes">
<option value="add">Add to existing file</option>
<option value="delete">Delete file</option>
<option value="read">Read existing file</option>
</cfif>
</select>
<input type="Hidden" name="formsubmit" value="yes">
<input type="Submit" name="" value="make my changes">
</form> --->

© 2014 Adobe Systems Incorporated. All rights reserved.
**cfile action = "append"**

**Description**

Appends text to a text file on the server.

**Syntax**

```xml
<cffile action = "append"
    file = "full pathname"
    output = "string"
    addNewLine = "yes|no"
    attributes = "file attributes list"
    charset = "characterset option"
    fixnewline = "yes|no"
    mode = "mode">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfdirectory

**History**

See the History section of the main `cfile` tag page.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
<tr>
<td>file</td>
<td>Required</td>
<td></td>
<td>Pathname of the file to which to append content of <code>output</code> attribute. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the <code>GetTempDirectory</code> function.</td>
</tr>
<tr>
<td>output</td>
<td>Required</td>
<td>String to append to the file.</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>----------</td>
<td>------------------------------</td>
<td></td>
</tr>
<tr>
<td>addNewLine</td>
<td>Optional</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: appends newline character to text written to file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no</td>
<td></td>
</tr>
<tr>
<td>attributes</td>
<td>Optional</td>
<td>Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify <code>attributes = &quot;readOnly&quot;</code>, all other attributes are overwritten.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• readOnly</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• hidden</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• normal</td>
<td></td>
</tr>
<tr>
<td>charset</td>
<td>Optional</td>
<td>JVM default file character set</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The character encoding in which the file contents is encoded. The following list includes commonly used values:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• utf-8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• iso-8859-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• windows-1252</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• us-ascii</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• shift_jis</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• iso-2022-jp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• euc-jp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• euc-kr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• big5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• euc-cn</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• utf-16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information character encodings, see <a href="http://www.w3.org/International/O-charset.html">www.w3.org/International/O-charset.html</a>.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| fixnewline   | Optional                                                                    | • yes: changes embedded line-ending characters in string variables to operating-system specific line endings  
|              |                                                                             | • no: (default) do not change embedded line-ending characters in string variables. For an example that uses this attribute, see `cffile action = "write"`. |
| mode         | Optional                                                                    | Applies only to UNIX and Linux. Permissions. Octal values of UNIX `chmod` command. Assigned to owner, group, and other, respectively; for example:  
|              |                                                                             | • 644: assigns read/write permission to owner; read permission to group and other.  
|              |                                                                             | • 777: assigns read/write/execute permission to all. |

Example

```xml
<!--The first example creates the file \temp\foo on a windows system and sets attributes to normal. --->
<cffile action = "write" file = "\temp\foo" attributes = normal output = "some text">  
  <!--- The second example appends to the file. --->
  <cffile action = "append" file = "\temp\foo" attributes = normal output = "Is this a test?">  
```
**cfile action = "copy"**

**Description**

Copies a file from one directory to another on the server.

**Syntax**

```cfile
<cffile
    action = "copy"
    destination = "full pathname"
    source = "full pathname"
    attributes = "file attributes list"
    mode = "mode">
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfdirectory

**History**

See the History section of the main cffile tag page.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
<tr>
<td>destination</td>
<td>Required</td>
<td></td>
<td>Pathname of a directory or file on web server where the file is copied. If you specify a filename without a directory path, ColdFusion copies it relative to the source directory.</td>
</tr>
<tr>
<td>Source</td>
<td>Required</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><strong>source</strong></td>
<td>Required</td>
<td>Pathname of the file to copy. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the <a href="https://help.adobe.com/cq/en/coldfusion-efficiency/Documentation/coldfusion/coldfusion-gettempdirectory.html">GetTempDirectory</a> function.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>attributes</strong></td>
<td>Optional</td>
<td>Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify <code>attributes = &quot;readOnly&quot;</code>, all other attributes are overwritten.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>readOnly</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>hidden</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>normal</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>mode</strong></td>
<td>Optional</td>
<td>Applies only to UNIX and Linux. Permissions. Octal values of UNIX chmod command. Assigned to owner, group, and other, respectively; for example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 644: assigns read/write permission to owner; read permission to group and other.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 777: assigns read/write/execute permission to all.</td>
</tr>
</tbody>
</table>

**Example**

This example copies the keymemo.doc file to the `c:\files\backup\` directory:

```
<cffile action = "copy" source = "c:\files\upload\keymemo.doc" destination = "c:\files\backup\">
```
**cfile action = "delete"**

**Description**

Deletes a file on the server.

**Syntax**

```cfile
  action = "delete"
  file = "full pathname"
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfdirectory

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
<tr>
<td>file</td>
<td>Required</td>
<td></td>
<td>Pathname of the file to delete. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the <code>GetTempDirectory</code> function.</td>
</tr>
</tbody>
</table>

**Example**

The following example deletes the specified file:

```cfile
  <cfile action = "delete"
    file = "c:\files\upload\#Variables.DeleteFileName#"
  >
```
cfile action = "move"

Description

Moves a file from one location to another on the server.

Syntax

```cfcuggage
<cffile
    action = "move"
    destination = "full pathname"
    source = "full pathname"
    attributes = "file attributes list"
    charset = "character set option"
    mode = "mode">
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfdirectory

History

See the History section of the main cfile tag page.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
<tr>
<td>destination</td>
<td>Required</td>
<td></td>
<td>Pathname of the destination directory or file. If not an absolute path, it is relative to the source directory.</td>
</tr>
<tr>
<td><strong>source</strong></td>
<td><strong>Required</strong></td>
<td><strong>Pathname of the file to move. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the <a href="#">GetTempDirectory</a> function.</strong></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| **attributes** | **Optional** | **Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify `attributes = "readOnly"`, all other attributes are overwritten.**  
  - `readOnly`  
  - `hidden`  
  - `normal` |
| **charset** | **Optional** | **JVM default file character set**  
  **The character encoding in which the file contents is encoded. The following list includes commonly used values:**  
  - `utf-8`  
  - `iso-8859-1`  
  - `windows-1252`  
  - `us-ascii`  
  - `shift_jis`  
  - `iso-2022-jp`  
  - `euc-jp`  
  - `euc-kr`  
  - `big5`  
  - `euc-cn`  
  - `utf-16`  
  **For more information character encodings, see [www.w3.org/International/O-charset.html](http://www.w3.org/International/O-charset.html).** |
**mode**  
Optional

Applies only to UNIX and Linux. Permissions. Octal values of UNIX chmod command. Assigned to owner, group, and other, respectively; for example:

- 644: assigns read/write permission to owner; read permission to group and other.
- 777: assigns read/write/execute permission to all.

### Example

The following example moves the keymemo.doc file from the `c:\files\upload\` directory to the `c:\files\memo\` directory in Windows:

```cfml
<cffile
  action = "move"
  source = "c:\files\upload\keymemo.doc"
  destination = "c:\files\memo\">
```

In this example, the destination directory is "memo."
cfile action = "read"

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

Description

Reads a text file on the server. The file is read into a dynamic, local variable that you can use in the page. For example:

- Read a text file; insert the file's contents into a database
- Read a text file; use the find and replace function to modify the file's contents

**Note**

This action reads the file into a variable in the local Variables scope. It is not intended for use with large files, such as logs, because this can bring down the server.

Syntax

```cfile
<cffile
  action = "read"
  file = "full pathname"
  variable = "variable name"
  charset = "character set option">
```

See also

cfdirectory

History

See the History section of the main cfile tag page.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
</tbody>
</table>
###cffile###

| **file** | **Required** | **Pathname of the file to read.**  
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the <strong>GetTempDirectory</strong> function.</td>
</tr>
<tr>
<td><strong>variable</strong></td>
<td><strong>Required</strong></td>
<td><strong>Name of variable to contain contents of text file.</strong></td>
</tr>
</tbody>
</table>
| **charset** | **Optional** | **Character encoding identified by the file's byte order mark, if any; otherwise, JVM default file character set.**  
|  |  | **The character encoding in which the file contents is encoded. The following list includes commonly used values:**  
|  |  | • utf-8  
|  |  | • iso-8859-1  
|  |  | • windows-1252  
|  |  | • us-ascii  
|  |  | • shift_jis  
|  |  | • iso-2022-jp  
|  |  | • euc-jp  
|  |  | • euc-kr  
|  |  | • big5  
|  |  | • euc-cn  
|  |  | • utf-16  
|  |  | If the file starts with a byte order mark and you set this attribute to a conflicting character encoding, ColdFusion generates an error. For more information character encodings, see [www.w3.org/International/O-charset.html](http://www.w3.org/International/O-charset.html). |

###Usage###

The following example creates a variable named Message for the contents of the file message.txt:

```cfc
cffile action = "read"  
file = "c:\web\message.txt"  
variable = "Message"
```
The variable \texttt{Message} can be used in the page. For example, you could display the contents of the message.txt file in the final web page as follows:

\begin{verbatim}
<cfoutput>#{Message}#</cfoutput>
\end{verbatim}

ColdFusion supports functions for manipulating the contents of text files. You can also use the variable that is created by a \texttt{cffile action = \textasciitilde{read}} operation in the \texttt{ArrayToList} and \texttt{ListToArray} functions.

\begin{itemize}
\item \textbf{Note}
\end{itemize}

If you use this tag to read a file that is encoded using the Windows Cp1252 (windows-1252) encoding of the Latin-1 character set on a system whose default character encoding is Cp1252, and the files has characters encoded in the Hex 8x or 9x range, specify \texttt{charset=\textasciitilde{windows-1252}} attribute, even though this is the default encoding. Otherwise, some characters in the Hex8x and 9x ranges that do not map correctly and display incorrectly.
cfile action = "readBinary"

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

Description

Reads a binary file (such as an executable or image file) on the server, into a binary object parameter that you can use in the page. To send it through a web protocol (such as HTTP or SMTP) or store it in a database, first convert it to Base64 using the ToBase64 function.

Note

This action reads the file into a variable in the local Variables scope. It is not intended for use with large files, such as logs, because they can bring down the server.

Syntax

```
<cfile
    action = "readBinary"
    file = "full pathname"
    variable = "variable name">
```

See also

cfdirectory

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
<tr>
<td>file</td>
<td>Required</td>
<td></td>
<td>Pathname of a binary file to read. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the GetTempDirectory function.</td>
</tr>
</tbody>
</table>
variable | Required | Name of variable to contain contents of binary file.
--- | --- | ---

Usage

You convert the binary file to Base64 to transfer it to another site. ColdFusion supports reading an image file as a binary and passing the result to a `cfimage`, for example:

```cfml
<!--- Convert a JPG image to a binary object. --->
<cffile action="readBinary" file="maxwell05.jpg" variable="binaryObject">
<!--- Create a cfimage from the binary object variable. --->
<cfset myImage=ImageNew(binaryObject)>
```

Example

The following example reads the binary file `somewhere.jpg`, writes it to a different folder as `somewhereB.jpg`, and then displays the new file:

```cfml
<cffile action = "readBinary" file = 
"C:\inetpub\wwwroot\cfdocs\getting_started\photos\somewhere.jpg" variable = 
"aBinaryObj">

<!--- Output binary object to JPEG format for viewing. --->
<cffile action="write" file = "c:\files\updates\somewhereB.jpg"  
output = 
"#toBinary(aBinaryObj)#">

<!--- HTML to view image. --->
<img src="C:\files\updates\somewhereB.jpg">
```
cfile action = "rename"

Description

Renames or moves a file on the server.

Syntax

```
<cffile
    action = "rename"
    destination = "pathname"
    source = "full pathname"
    attributes = "file attributes list"
    mode = "mode">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

See also

cfdirectory

History

See the History section of the main cfile tag page.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
<tr>
<td>destination</td>
<td>Required</td>
<td></td>
<td>Destination file or directory. If not an absolute path, it is relative to the source directory.</td>
</tr>
<tr>
<td>source</td>
<td>Required</td>
<td></td>
<td>Pathname of file to rename. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the GetTemp Directory function.</td>
</tr>
</tbody>
</table>
### attributes

Optional

Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if `attributes = "readOnly"`, all other attributes are overwritten.

- `readOnly`
- `hidden`
- `normal`

### mode

Optional

Applies only to UNIX and Linux. Permissions. Octal values of UNIX `chmod` command. Assigned to owner, group, and other. For example:

- `644`: assigns read/write permission to owner; read permission to group and other.
- `777`: assigns read/write/execute permission to all.

## Usage

The `rename` action renames or move a file. The `destination` attribute must be a pathname, not just a new name for the file. If the destination is a directory, the file is moved and not renamed.

### Example

#### Windows example:

```c
c<!--- Source Document is read-only but when renamed it becomes normal (not hidden or read-only). --->
c<cffile action = "rename" source = "c:\files\memo\readonlymemo.doc" destination = "c:\files\memo\normalmemo.doc" attributes="normal">
```

#### UNIX example:

...
<cffile action = "rename" source = "#myWR#/memo/sample.txt"
destination = "#myWR#/memo/other_sample.txt" mode="666"/>
**cffile action = "upload"**

**Description**

Copies a file to a directory on the server.

**Syntax**

```xml
<cffile
action = "upload"
destination = "full pathname"
fileField = "form field"
accept = "MIME type|file type"
attributes = "file attribute or list"
mode = "permission"
nameConflict = "behavior"
result = "result name"
ContinueOnError = "true|false"
Errors = " variable in which the file upload errors will be stored">
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

`cfdirectory`, `cffile action = "uploadAll"`, `cffileupload`

**History**

The strict attribute has been added in ColdFusion 10. See the History section of the main `cffile` tag page.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>destination</td>
<td>Required</td>
<td>Pathname of directory in which to upload the file. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the <code>GetTempDirectory</code> function. If the destination you specify does not exist, ColdFusion creates a file with the specified destination name. For example, if you specify the destination, C:\XYZ, ColdFusion creates a file XYZ in the C: drive.</td>
<td></td>
</tr>
<tr>
<td>fileField</td>
<td>Optional</td>
<td>Name of form field used to select the file. Do not use number signs (#) to specify the field name. If omitted, it defaults to the name of the first file field submitted.</td>
<td></td>
</tr>
<tr>
<td>accept</td>
<td>Optional</td>
<td>Limits the MIME types to accept. Comma-delimited list. For example, the following code permits JPEG and Microsoft Word file uploads: <code>accept = {&quot;image/jpg, application/msword&quot;}</code> The browser uses the file extension to determine file type.</td>
<td></td>
</tr>
</tbody>
</table>
| attributes | Optional | Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify `attributes = "readOnly"`, all other attributes are overwritten.  
- `readOnly`
- `hidden`
- `normal` (if you use this option with other attributes, it is overridden by them) |
| --- | --- | --- |
| mode | Optional | Applies only to UNIX and Linux. Permissions. Octal values of `chmod` command. Assigned to owner, group, and other, respectively, for example:  
- 644: assigns read/write permission to owner; read permission to group and other.  
- 777: assigns read/write/execute permission to all. |
<table>
<thead>
<tr>
<th>nameConflict</th>
<th>Optional</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action to take if filename is the same as that of a file in the directory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Error: file is not saved. ColdFusion stops processing the page and returns an error.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Skip: file is not saved. This option permits custom behavior based on file properties.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Overwrite: replaces file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• MakeUnique: forms a unique filename for the upload; name is stored in the file object variable serverFile.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>result</th>
<th>Optional</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lets you specify a name for the variable in which cffile returns the result (or status) parameters. If you do not specify a value for this attribute, cffile uses the prefix cffile. For more information, see Usage.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>strict</th>
<th>Optional</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>When strict is true, only MIME types or a combination of MIME types and extensions are allowed in the accept attribute. Since strict is true by default, you should specify MIME types for the accept attribute. When strict is false, either MIME types or extensions or a combination of both can be specified as a value to the accept attribute. For more information, see this blog entry.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| ContinueOnError | Optional | False | By default, when uploading one of the files fail, the remaining files will not be uploaded. If this value is set to true, file upload continues even after encountering an upload error. A file upload error happens due to the following reasons:

1. Empty file
2. Invalid file type
3. Invalid MIME or extension
4. File already exists

In the case of an upload failure, the error details will be stored in the errors attribute. |
|---|---|---|---|
| Errors | Optional | cffile.uploadAllErrors | The name of the variable in which the file upload errors will be stored. Errors will be populated in the specified variable name when continueOnError is true. After the file upload is completed, this tag creates an array of structures that contains upload failure information for each upload failure.

The upload failure information error structure contains the following fields: |
Usage

After a file upload is completed, you can get status information using file upload parameters. To refer to parameters,
use either the cffile prefix or, if you specified an alternate name in the result attribute, the name you specified there. For example, if you did not specify a name in the result attribute, access the fileExisted parameter as #cffile.fileExisted#. If you set the result attribute to myResult, however, access fileExisted as #myResult.fileExisted#. Status parameters can be used anywhere that other ColdFusion parameters can be used. When you use a cfform tag or an HTML form tag to submit the form with the file to be uploaded, specify enctype = "multipart/form-data" in the tag, as shown in the example for this tag. By default, ColdFusion sends the form with the encoding type of application/x-www-form-urlencoded, which causes an error in the cffile tag.

The result attribute allows functions or CFCs that get called from multiple pages at the same time to avoid overwriting the results of one call with another.

Note

The file prefix is deprecated, in favor of the cffile prefix. Do not use the file prefix in new applications.

If your page is uploading a file that was selected on a form or was otherwise sent to your page via a multipart/form-data HTTP message, you can determine the approximate size of the file by checking the value of the CGI.content_length variable. This variable includes the file length plus the length of any other request content.

The following file upload status parameters are available after an upload:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attemptedServerFile</td>
<td>Initial name ColdFusion used when attempting to save a file</td>
</tr>
<tr>
<td>clientDirectory</td>
<td>Directory location of the file uploaded from the client's system</td>
</tr>
<tr>
<td>clientFile</td>
<td>Name of the file uploaded from the client's system</td>
</tr>
<tr>
<td>clientFileExt</td>
<td>Extension of the uploaded file on the client system (without a period)</td>
</tr>
<tr>
<td>clientFileName</td>
<td>Name of the uploaded file on the client system (without an extension)</td>
</tr>
<tr>
<td>contentSubType</td>
<td>MIME content subtype of the saved file</td>
</tr>
<tr>
<td>contentType</td>
<td>MIME content type of the saved file</td>
</tr>
<tr>
<td>dateLastAccessed</td>
<td>Date and time the uploaded file was last accessed</td>
</tr>
<tr>
<td>fileExisted</td>
<td>Whether the file existed with the same path (yes or no)</td>
</tr>
<tr>
<td>fileSize</td>
<td>Size of the uploaded file in bytes</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>fileWasAppended</td>
<td>Whether ColdFusion appended uploaded file to a file (yes or no)</td>
</tr>
<tr>
<td>fileWasOverwritten</td>
<td>Whether ColdFusion overwrote a file (yes or no)</td>
</tr>
<tr>
<td>fileWasRenamed</td>
<td>Whether uploaded file renamed to avoid a name conflict (yes or no)</td>
</tr>
<tr>
<td>fileWasSaved</td>
<td>Whether ColdFusion saves a file (yes or no)</td>
</tr>
<tr>
<td>oldFileSize</td>
<td>Size of a file that was overwritten in the file upload operation</td>
</tr>
<tr>
<td>serverDirectory</td>
<td>Directory of the file saved on the server</td>
</tr>
<tr>
<td>serverFile</td>
<td>Filename of the file saved on the server</td>
</tr>
<tr>
<td>serverFileExt</td>
<td>Extension of the uploaded file on the server (without a period)</td>
</tr>
<tr>
<td>serverFileName</td>
<td>Name of the uploaded file on the server (without an extension)</td>
</tr>
<tr>
<td>timeCreated</td>
<td>Time the uploaded file was created</td>
</tr>
<tr>
<td>timeLastModified</td>
<td>Date and time of the last modification to the uploaded file</td>
</tr>
</tbody>
</table>

**Note**

File status parameters are read-only. They are set to the results of the most recent `cffile` operation. If two `cffile` tags execute, the results of the second overwrite the first, unless you have specified a different result variable in the `result` attribute.

**Example**

The following example creates a unique filename, if there is a name conflict when the file is uploaded on Windows:
<!-- Windows Example --->
<!-- Check to see if the Form variable exists. --->
<cfif isDefined("Form.FileContents")>
<!-- If TRUE, upload the file. --->
<cffile action = "upload"
    fileField = "FileContents"
    destination = "c:\files\upload\"
    accept = "text/html"
    nameConflict = "MakeUnique">
<cfelse>
<!-- If FALSE, show the Form. --->
<form method="post" action=#cgi.script_name#>
    name="uploadForm" enctype="multipart/form-data">
    <input name="FileContents" type="file">
    <br>
    <input name="submit" type="submit" value="Upload File">
</form>
</cfif>
cfile action = "uploadAll"

Description

Copies all files sent to the page in an HTTP request to a directory on the server.

Syntax

```
<cffile
   action = "uploadAll"
   destination = "full pathname"
   accept = "list of MIME types"
   attributes = "file attribute or list"
   mode = "permission"
   nameConflict = "behavior"
   result = "result name"
   ContinueOnError = "true|false"
   Errors = Variable in which the file upload errors will be stored."
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfile action = "upload", cfdirectory

History

ColdFusion 11: Added the attributes ContinueOnError and Errors.

See the History section of the main cfile tag page.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
<tr>
<td><strong>destination</strong></td>
<td>Required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pathname of directory in which to upload the file. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the GetTempDirectory function.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>accept</strong></th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limits the MIME types to accept. Comma-delimited list. For example, the following code permits JPEG and Microsoft Word file uploads:accept = {&quot;image/jpg, application/msword&quot;}The browser uses the file extension to determine file type.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>attributes</strong></th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify attributes = &quot;readOnly&quot;, all other attributes are overwritten.</td>
<td></td>
</tr>
<tr>
<td>• readOnly</td>
<td></td>
</tr>
<tr>
<td>• hidden</td>
<td></td>
</tr>
<tr>
<td>• normal (if you use this option with other attributes, it is overridden by them)</td>
<td></td>
</tr>
<tr>
<td><strong>mode</strong></td>
<td>Optional</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>nameConflict</strong></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ContinueOnError</td>
<td>Optional</td>
</tr>
<tr>
<td>Errors</td>
<td>Optional</td>
</tr>
</tbody>
</table>
- **REASON** - The reason for the failure
- **DETAIL** - File upload failure detail
- **MESSAGE** - A detailed message depicting the failure
- **CLIENTFILE** - Name of the file uploaded from the client's system
- **CLIENTFILEEXT** - Extension of the uploaded file on the client system (without a period)
- **CLIENTFILENAME** - Name of the uploaded file on the client system (without an extension)
- **INVALID_FILE_TYPE** - If the file mime type or extension is not in the specified accept attribute. If the reason is INVALID_FILE_TYPE, two additional keys will be available in the structure.
  - **ACCEPT**: list of mime types or file extensions given in the tag
  - **MIMETYPE**: mime type of the uploaded file
- **EMPTY_FILE** - If the uploaded file is an empty file
- **FILE_EXISTS** - If any file with the given name already exists in the destination and the overwrite policy is error.
- **DEST** - The destination where file is copied
- **FORM_FILE_NOT_FOUND** - If the uploaded file is not found on the server
result | Optional | Lets you specify a name for the variable in which cffile returns the result (or status) parameters. If you do not specify a value for this attribute, cffile uses the prefix cffile. For more information, see Usage.

Usage

Unlike cffile action="upload", which uploads only one file at a time, cffile action="uploadall" uploads multiple files thereby eliminating the need to code multiple cffile action="upload" statements. Use this tag in the page specified by the action attribute of a cffileupload control. This tag uploads save the files that the cffileupload control sends when the user clicks the Save File button. After a file upload is completed, this tag creates an array of structures specified by the result parameter. Each structure in the array contains upload result information for one file. For information on the result structure contents, see cffile action = "upload".

⚠️ Note

You can control the maximum file size of the upload by specifying the server Request Throttle Threshold or the Settings page of the Administrator Server Settings section.

Example

The following example copies files uploaded by a cffileupload tag to a temp directory.

```cfml
<cfif isDefined("form.submit")>
  <cffile action="uploadall" destination="#expandpath('./upload')#">
</cfif>
<cfform action="#cgi.script_name#" enctype="multipart/form-data">
  <cfinput type="file" name="attachment1">
  <cfinput type="file" name="attachment2">
  <cfinput type="file" name="attachment3">
  <cfinput type="submit" name="submit" value="submit">
</cfform>
```
**cfile action = "write"**

**Description**

Writes a text file on the server, based on dynamic content. You can create static HTML files from the content, or log actions in a text file.

**Syntax**

```xml
<cffile
  action = "write"
  file = "full pathname"
  output = "content"
  addNewLine = "yes|no"
  attributes = "file attributes list"
  charset = "character set option"
  fixnewline = "yes|no"
  mode = "permission">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfdirectory

**History**

See the History section of the main cfile tag page.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Type of file manipulation that the tag performs.</td>
</tr>
<tr>
<td>file</td>
<td>Required</td>
<td></td>
<td>Pathname of the file to write. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the GetTempDirectory function.</td>
</tr>
<tr>
<td><strong>output</strong></td>
<td><strong>Required</strong></td>
<td><strong>Content of the file to be created.</strong></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>--------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>addNewLine</strong></td>
<td><strong>Optional</strong></td>
<td><strong>yes</strong></td>
<td></td>
</tr>
</tbody>
</table>
|               |             | - yes: appends newline character to text written to file.  
|               |             | - no                                  |
| **attributes**| **Optional** | **Applies to Windows. A comma-delimited list of attributes to set on the file. If omitted, the file's attributes are maintained. Each value must be specified explicitly. For example, if you specify attributes = "readOnly", all other attributes are overwritten.** |
|               |             | - readOnly  
|               |             | - hidden  
|               |             | - normal  |
| **charset**   | **Optional** | **JVM default file character set** |
|               |             | **The character encoding in which the file contents is encoded. The following list includes commonly used values:** |
|               |             | - utf-8  
|               |             | - iso-8859-1  
|               |             | - windows-1252  
|               |             | - us-ascii  
|               |             | - shift_jis  
|               |             | - iso-2022-jp  
|               |             | - euc-jp  
|               |             | - euc-kr  
|               |             | - big5  
|               |             | - euc-cn  
|               |             | - utf-16  
<p>|               |             | For more information character encodings, see <a href="http://www.w3.org/International/O-charset.html">www.w3.org/International/O-charset.html</a>. |</p>
<table>
<thead>
<tr>
<th>fixnewline</th>
<th>Optional</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td>• yes: changes embedded line-ending characters in string variables to operating-system specific line endings.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• no: does not change embedded line-ending characters in string variables.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mode</th>
<th>Optional</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies only to UNIX and Linux. Permissions. Octal values of UNIX chmod command. Assigned to owner, group, and other, respectively; for example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 644: assigns read/write permission to owner; read permission to group and other.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 777: assigns read/write/execute permission to all.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

This example creates a file with information a user entered in an HTML insert form:

```cdata
<cffile action = "write"
  file = "c:\files\updates\#Form.UpdateTitle#.txt"
  output = "Created By: #Form.FullName#
          Date: #Form.Date#
          #Form.Content#">
```

If the user submitted a form with the following:

```cdata
UpdateTitle = "FieldWork"
FullName = "World B. Frueh"
Date = "10/30/01"
Content = "We had a wonderful time in Cambridgeport."
```

ColdFusion would create a file named FieldWork.txt in the c:\files\updates\ directory and the file would contain the following text:
We had a wonderful time in Cambridgeport.

This example shows the use of the `mode` attribute for UNIX. It creates the file `/tmp/foo` with permissions `rw-r-r-` (owner = read/write, group = read, other = read):

```
<cffile action = "write"
       file = "/tmp/foo"
       mode = 644>
```

This example appends to the file and sets permissions to read/write (rw) for all:

```
<cffile action = "append"
       destination = "/home/tomj/testing.txt"
       mode = 666
       output = "Is this a test?"
```

This example uploads a file and gives it the permissions owner/group/other = read/write/execute):

```
cffile action = "upload"
fileField = "fieldname"
destination = "/tmp/program.exe"
mode = 777
```

This example uses the `fixnewline` attribute to changes embedded line-ending characters in `xmlString`, which is derived from `xmlData`, to operating-system specific line endings.
ColdFusion supports using `cffile` to write an image, for example:

```coldfusion
<!--- Create a new cfimage. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Draw a square on the image. --->
<cfset ImageDrawRect(myImage,10,10,100,100)>
<!--- Use cffile to write the cfimage to a JPG. --->
<cffile action="write" output="#myImage#" file="c:\cfpix\square.jpg">
```
cfupload

Description
Displays a dialog for uploading multiple files from the user's system. The enhanced dialog includes the following features:

- You can specify the maximum number of files and the maximum file size for the upload.
- A progress bar to visually indicate the overall progress of the file upload task, and another progress bar to indicate the upload progress of each individual file.
- A success or failure message appears for each file upload and the overall upload task.
- At any point during the upload task, you can cancel the upload.

Category
File management tags, Forms tags

Syntax

```<cfupload>
  addbuttonlabel= "label"
  align = align="center|left|right"
  bgcolor = "color"
  clearbuttonlabel = "label"
  deletebuttonlabel = "label"
  extensionfilter = "none|jpg,jpeg,png"
  height= "number of pixels"
  hideUploadButton = "true|false"
  maxfileselect = "number of files"
  maxuploadsize = "file size in mega bytes"
  name = "File uploader name"
  oncomplete = "JavaScript function name"
  onerror = "JavaScript function name"
  onUploadComplete = "JavaScript function name"
  progressbar = "true|false"
  stoponerror = "true|false"
  style = "style specification"
  title = "Title panel name"
  uploadbuttonlabel = "label"
  url = "URL"
  width = "number of pixels"
  wmode = "window|opaque|transparent"
</cfupload>```

Note
You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cf file action = "uploadAll"

History
ColdFusion 9: Added this tag.

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addbuttonlabel</td>
<td>Optional</td>
<td>Add Files</td>
<td>Label of the Add button.</td>
</tr>
</tbody>
</table>
| align                  | Optional| left    | Specifies the default alignment. The following values are valid:  
|                        |         |         |   - center  
|                        |         |         |   - left    
|                        |         |         |   - right   |
| bgcolor                | Optional|         | The background color for the file upload control. A hexadecimal value without "#" prefixed or a recognized color name, for example red. |
| clearbuttonlabel       | Optional| Clear All | Label of the Clear button |
| deletebuttonlabel      | Optional| Delete  | Label of the Delete button |
| extensionfilter        | Optional| none    | Use this attribute to specify the type of file that you will allow to be uploaded. For example, to let only image files to be uploaded, you can specify file extensions such as .jpg, .jpeg, or .png. If set to none, files are uploaded without any extension filter. |
| height                 | Optional| 300     | Height of the file upload control, in pixels. |
| hideUploadButton       | Optional| false   | A Boolean value that specifies if the Upload button should appear in the file upload dialog:  
|                        |         |         |   - true  
<p>|                        |         |         |   - false |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxfileselect</td>
<td>Optional</td>
<td></td>
<td>The maximum number of files allowed for upload.</td>
</tr>
<tr>
<td>maxuploadsize</td>
<td>Optional</td>
<td>10MB</td>
<td>The maximum file size, in Megabytes, allowed for upload in an operation. ColdFusion throws an error if the value of the attribute maxuploadsize exceeds the throttle/post data size settings specified in the ColdFusion Administrator.</td>
</tr>
<tr>
<td>name</td>
<td>Optional</td>
<td></td>
<td>Name of the file upload component.</td>
</tr>
</tbody>
</table>
| onComplete      | Optional |         | The JavaScript function to run when a file has finished uploading. By default, ColdFusion passes a JavaScript object as a parameter to this function with the following properties:  
- **STATUS** - numeric value that is based on the HTTP status code  
- **MESSAGE** - Passed or Failed  
- **FILENAME** - Name of the file selected for upload  
You can also pass the JavaScript object by creating a struct with parameters "status" and "message" and call serializeJSON() on the JavaScript object. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| onError       | Optional   | The JavaScript function to run if the uploading of a file fails. The error can be a network error or server-side error. By default, ColdFusion passes a JavaScript object as a parameter to this function with the following properties:  
  - STATUS - numeric value that is based on the HTTP status code  
  - MESSAGE - Passed or Failed  
  - FILENAME - Name of the file selected for upload  
  You can also pass the JavaScript object by creating a struct with parameters "status" and "message" and call serializeJSON() on the JavaScript object. |
| onUploadComplete | Optional | The JavaScript function to run after uploading all the files.                                                                                     |
| progressbar   | Optional   | true  
  Whether to display a progress bar while the files upload:  
  - true  
  - false  |
| stopOnError   | Optional   | true  
  Specifies whether to ignore the exceptions for this operation.  
  - true - Stops uploading and displays an appropriate error.  
  - false - Continues uploading and displays an appropriate error. |
<p>| style         | Optional   | A CSS style specification that defines layout styles.                                                                                          |</p>
<table>
<thead>
<tr>
<th>attribute</th>
<th>optional</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>Optional</td>
<td></td>
<td>Title for the upload dialog.</td>
</tr>
<tr>
<td>uploadbuttonlabel</td>
<td>Optional</td>
<td>Upload</td>
<td>Label of the Upload button.</td>
</tr>
<tr>
<td>url</td>
<td>Optional</td>
<td></td>
<td>The URL to the server where the files are uploaded. The attribute is optional and it defaults to cgi.script_name.</td>
</tr>
<tr>
<td>width</td>
<td>Optional</td>
<td>420</td>
<td>Width of the file upload control, in pixels.</td>
</tr>
<tr>
<td>wmode</td>
<td>Optional</td>
<td>window</td>
<td>Specifies the absolute positioning and layering capabilities in your browser:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• window: Plays the media player in its own rectangular window on a web page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• opaque: Hides everything behind the media player on the web page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• transparent: Lets the background of the web page show through the transparent portions of the media player</td>
</tr>
</tbody>
</table>

**Usage**

Use this tag to create a SWF file-based file upload control that lets a user upload multiple files to a server. To upload files to the server, define a server-side template. The template that you define reads the upload request and uploads the selected files to the server.

**Enhancements made in ColdFusion 9.0.1**

In ColdFusion 9.0.1, the fileupload control passes the session information implicitly to the target page if session management is turned on either in Application.cfc or Application.cfm. Assume that fileupload control is defined without a URL attribute. In this case, if the user chooses to upload data using the upload button, the control comes back to the same page. Users can check for form.fieldnames to perform the upload as shown in the following example: Upload.cfm
<cfif isdefined("form.FIELDNAMES")>
  <cffile action = "upload" destination = "#ExpandPath('.')#"
        nameconflict="makeunique">
</cfif>
<cffileupload name="myuploader">

In this case, url defaults to CGI.script_name.
To maintain sessions between the fileupload control and the URL, users must turn on session management. You can do this by setting the this.sessionmanagement=true in Application.cfc. The setting ensures that CFID and CFtoken are passed as part of the URL if Enable J2EE Session Variables (ColdFusion Administrator > Server Settings > Memory Variables) is not selected. If it is selected, then JsessionID is passed as part of the URL.

Supported Styles

The following are the supported styles:

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headercolors</td>
<td>Format: color; colors of the band at the top of the DateChooser control. Specify two values, separated by a comma. For a solid band, use the same color for both values. The default value is ##E6EEEE,##FFFFFF.</td>
</tr>
<tr>
<td>textcolor</td>
<td>Color of text. Can be a hexadecimal value or a named color. For a hexadecimal value, use the form &quot;##xxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none.</td>
</tr>
<tr>
<td>titletextalign</td>
<td>Aligns the title text. The recognized values are left, right, and center. The default value is right.</td>
</tr>
<tr>
<td>titletextcolor</td>
<td>Color of the title text.</td>
</tr>
<tr>
<td>bgcolor</td>
<td>The background color for the file upload control. A hexadecimal value without &quot;#&quot; prefixed or a recognized color name, for example red.</td>
</tr>
<tr>
<td>rollovercolor</td>
<td>Displays values on mouse-over.</td>
</tr>
<tr>
<td>selectcolor</td>
<td>Background color for a selected item. Can be a hexadecimal value or a named color. For a hexadecimal value, use the form &quot;##xxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none. For a list of the supported named colors, see cfchart.</td>
</tr>
</tbody>
</table>

Example
<h3>Instructions</h3>
<p>Create a folder Upload in your C: drive<br>Try uploading files using the file upload component and check if the files have<br>been appropriately saved in the Upload folder.</p>

```html
<script>
var foo = function(result)
{
  alert(ColdFusion.JSON.encode(result));
}
</script>
<cffileupload
url="uploadFiles.cfm"
progressbar="true"
name="myupload"
addButtonLabel = "Add File"
clearButtonlabel = "Clear it"
hideUploadButton = "true"
width=600
height=400
title = "File Upload"
maxuploadsize="30"
extensionfilter="*.jpg, *.png, *.flv, *.txt"
BGCOLOR="#FFFFFF"
MAXFILESELECT=10
UPLOADBUTTONLABEL="Upload now"/>
```

Uploadfiles.cfm is given below:

```html
<cffile action="upload" destination="#expandpath('./upload')#"
nameconflict="makeunique">
<cfoutput>#serializeJSON({STATUS=200,MESSAGE='Passed'})#</cfoutput>
```

This example sends user-specified files to the server-side template - uploadfiles.cfm. The template file that you define can use the "upload" or "uploadall" action defined in the cffile tag.

⚠️ **Note**

The filefield attribute of the upload action is optional.

Use the destination attribute in the cffile tag to define the location to save the files. For the uploadfiles.cfm code, see `cffile action = "uploadAll"`. 
**cfinally**

**Description**

Used inside a `cftry` tag. Code in the `cfinally` block is processed after the main `cftry` code and, if an exception occurs, the `cfcatch` code. The `cfinally` block code always executes, whether or not there is an exception.

**Category**

Exception handling tags

**Syntax**

```plaintext
<cftry>
  try code  <cfcatch>
    catch code  <cfcatch>
    ...
  <cfinally>
    final code  </cfinally>
</cftry>
```

**See also**

`cftry`, `cfcatch`, `cferror`, `cfrethrow`, `cfthrow`, `onError`; [Handling Errors](https://developer.adobe.com/cfx/techref/) in the *Developing ColdFusion Applications*

**History**

ColdFusion 9: Added the tag

**Usage**

The `cfinally` tag is optional in a `cftry` block, and the block can have only one `cfinally` tag. Put the `cfinally` tag at the end of all `cftry` block, after any `cfcatch` blocks. This tag requires an end tag. You can nest `cftry/cfcatch/cfinally` blocks. Use the `cfinally` tag for code that should execute whether or not an exception occurs. For example, use it to free up resources.

**Example**
<h3>cffinally Example</h3>
<!--- Open a cftry block. --->
<cftry>
    ....
    <cfcatch type = "Database">
        ....
    </cfcatch>
    <cffinally>
    ....
    <!--- Do some cleanup here before leaving cftry block --->
    ....
    </cffinally
</cftry>
**cfflush**

**Description**

Flushes currently available data to the client.

**Category**

Data output tags, Page processing tags

**Syntax**

```html
<cfflush
    interval = "integer number of bytes">
```

**Note**

You can specify this tag's attributes in an attributeName attribute whose value is a structure. Specify the structure name in the attributeName attribute and use the tag's attribute names as structure keys.

**See also**

cfcache, cfheader, cfinclude, cfsetting, cfsilent

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval</td>
<td>Optional</td>
<td></td>
<td>Integer. Flushes output each time this number of bytes becomes available. HTML headers, and data that is already available when the tag is executed, are omitted from the count.</td>
</tr>
</tbody>
</table>

**Usage**

The first occurrence of this tag on a page sends back the HTML headers and any other available HTML. Subsequent cfflush tags on the page send only the output that was generated after the previous flush. When you flush data, ensure that enough information is available, as some browsers might not respond if you flush only a small amount. Similarly, set the interval attribute for a few hundred bytes or more, but not thousands of bytes. Use the interval attribute only when a large amount of output is sent to the client, such as in a cfloop or a cfoutput of a large query. Using this form globally (such as in the Application.cfm file) might cause unexpected errors when CFML tags that modify HTML headers are executed. Because the cfflush tag sends data to the browser when it executes, it has several limitations, including the following:

- Using any of the following tags or functions on a page anywhere after the cfflush tag can cause errors or unexpected results: cfcontent, cfcookie, cfform, cfheader, cfhtmlhead, cflocation, and SetLo
cale. Similarly, do not use any tags that use AJAX features, including `cfdiv`, `cflayout`, `cflayoutarea`, `cfpod`, `cfsprydataset`, `cftooltip`, `cfwindow`, or HTML format `cfgrid`, `cftree`, `cftextarea`, or `cfinput` (using `autosuggest` or `datefield` attributes) tags. All of the preceding tags and functions normally modify the HTML header, but cannot do so after a `cflush` tag, because the `cflush` sends the header.

- Using the `cfset` tag to set a cookie anywhere on a page that has a `cflush` tag does not set the cookie in the browser.
- Using the `cflush` tag in the body of several tags, including `cfsavecontent`, `cfquery`, and custom tags, causes errors.
- If you save Client variables as cookies, any client variables that you set after a `cflush` tag are not saved in the browser.

**Note**

Normally, the `cferror` tag discards the current output buffer and replaces it with the contents of the error page. The `cflush` tag discards the current buffer. As a result, the `Error.GeneratedContent` variable resulting from a `cferror` tag after a `cflush` contains any contents of the output buffer that has not been flushed. This content is not sent to the client. The content of the error page displays to the client after the bytes that have been sent.

**Example**

The following example uses `cfloop` tags and the `rand` random number generating function to delay data display. It simulates a page that is slow to generate data.

```cfml
<h1>Your Magic numbers</h1>
<p>It will take us a little while to calculate your ten magic numbers. It takes a lot of work to find numbers that truly fit your personality. So relax for a minute or so while we do the hard work for you.</p>
<h2>We are sure you will agree it was worth the short wait!</h2>
<cflush>
</cflush>
<cflush interval=10>
<!-- Delay Loop to make it seem harder. --->
<cfloop index="randomindex" from="1" to="200000" step="1">
   <cfset random=rand()>
</cfloop>
<!-- Now slowly output 10 random numbers. --->
<cfloop index="Myindex" from="1" to="10" step="1">
   <cfset random=rand()>
   <cfloop index="randomindex" from="1" to="100000" step="1">
      <cfset random=rand()>
   </cfloop>
   <cfoutput>
      Magic number #Myindex# is: &nbsp;&nbsp;#RandRange(100000, 999999)#<br>
   </cfoutput>
</cfloop>
</cflush>
</cfloop>
```
**cformgroup**

**Description**

Creates a container control for multiple form controls. Used in the `cform` tag body of Adobe Flash and XML forms. Ignored in HTML forms.

**Category**

*Forms tags*

**Syntax**

```html
<cfformgroup
    type = "group type"
    label = "label"
    style = "style specification"
    selectedIndex = "page number">
    width = "pixels"
    height = "pixels"
    enabled = "yes|no"
    visible = "yes|no"
    onChange = "ActionScript expression"
    tooltip = "text"
    id = "unique identifier">
    ...ColdFusion forms controls...
</cfformgroup>

OR

```html
<cfformgroup
    type = "repeater"
    query = "query object"
    maxrows = "integer">
    startrow = "row number"
    ...ColdFusion forms controls
</cfformgroup>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

`cfapplet`, `cfcalendar`, `cform`, `cformitem`, `cfgrid`, `cfinput`, `cfselect`, `cfslider`, `cftextarea`, and `cftree`

**History**

*ColdFusion MX 7: Added this tag.*

**Attributes**
The following table lists the attributes and their behavior in Flash forms. For XML, if not otherwise noted, the attribute is passed to the XML but is not interpreted by the basic XSL style sheet provided with ColdFusion.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt; formats</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Required; Flash and XML</td>
<td></td>
<td>XML: Can be any XForms group type defined in the XSLT. The XSL skins provided with ColdFusion support the following types:</td>
</tr>
</tbody>
</table>

- **horizontal**: align child tags horizontally within a form and put this tag's `label` attribute to the left of the children.
- **vertical**: align child tags vertically within a form and put this tag's `label` attribute to the left of the children.
- **fieldset**: corresponds to the HTML `fieldset` tag, which groups its children, typically by drawing a box around them and replacing part of the top line with legend text. To specify the legend, use the `label` attribute. To specify the box dimensions, use the `style` attribute with `height` and `width` values. Explicitly use `cfformgroup type="vertical"` inside this formgroup to align its child tags vertically.

**Note**

Attributes that are not marked as supported in XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.
Flash: Must be one of the following:

- **repeater**: dynamically creates an instance of the `cfformgroup`'s child tag or tags for each row of a query object, without requiring ColdFusion to recompile the Flash SWF file when the number of rows changes.
- **horizontal**: aligns child tags horizontally within a form and put this tag's `label` attribute to the left of the children. Use this tag to arrange individual controls horizontally.
- **vertical**: aligns child tags vertically within a form and puts this tag's `label` attribute to the left of the children. Use this tag to arrange individual controls vertically.
- **hbox**: aligns children horizontally. Use this type to arrange groups of form controls horizontally. Do not use this attribute to align individual controls horizontally, because the child controls do not align properly; use the horizontal type instead.
- **vbox**: aligns children vertically. Use this type to arrange groups of controls vertically. Do not use this attribute to align individual controls vertically, because the child controls do not align properly; use the vertical type instead.
hdividedbox: aligns children horizontally. Each child is in a box with a border, and there are dividers between the boxes that users can move to change the relative sizes of the children. Use a tag with this attribute to arrange groups of form controls horizontally. You cannot use this attribute to align individual controls horizontally.

vdividedbox: aligns children vertically. Each child is in a box with a border, and there are dividers between the boxes that users can move to change the relative sizes of the children. Use this type to group form controls, for example as a unit in an hbox form group. Do not use this attribute to align individual tags vertically.

panel: a container consisting of a title bar containing the label at tribute text, a border, and a content area with vertically arranged children.

tile: places the children in a rectangular grid.

accordion: places each child in a pleat of an expanding and contracting accordion. Define each pleat using a cfformgroup type="page" tag.
- **tabnavigator**: places the children in a tabbed dialog box. Define each tab by using a `cformgroup` tag type="page".

- **page**: places the children tags, aligned vertically, in a single tab of a parent tabnavigator or pleat of an accordion container.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>Required for type= repeater, ignored otherwise; Flash. The query to use with the repeater. Flash creates an instance of each of the <code>cformgroup</code> tag's child tags for each row in the query. You can use the bind attribute in the child tags to use data from the query row for the instance.</td>
</tr>
<tr>
<td>enabled</td>
<td>Optional; Flash. Boolean value that specifies whether the controls in the form group are enabled. Disabled controls appear in light gray.</td>
</tr>
<tr>
<td>height</td>
<td>Optional; Flash. Height of the group container, in pixels. If you omit this attribute, Flash automatically sizes the container height. Ignored for Flash repeater type.</td>
</tr>
<tr>
<td>id</td>
<td>Optional; Flash. Unique identifier for the form group. When using the tabnavigator or accordion type, specify the id attribute to reference the controls through custom ActionScript.</td>
</tr>
<tr>
<td>Field</td>
<td>Optional;Flash and XML</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------</td>
</tr>
</tbody>
</table>
| label        | Optional;Flash and XML | Label to apply to the form group. In Flash, does the following:  
- For a page or panel form group, determines the label to put on the corresponding accordion pleat, the tabnavigator tab, or the panel title bar. For a Flash horizontal or vertical form group, specifies the label to put to the left of the group.  
- Ignored in Flash for repeater, hbox, hdividedbox, vbox, vdividedbox, tile, accordion, and tabnavigator types. |
<p>| maxrows      | Optional;Flash         | Used only for the repeater type; ignored otherwise. Specifies the maximum number of query rows to use in the Flash form repeater. If the query has more rows than the sum of the startrow attribute and this value, the repeater does not use the remaining rows. |
| onChange     | Optional;Flash         | Tabnavigator and accordion types only: ActionScript expression or expressions to execute when a new tab or accordion page is selected. <strong>Note:</strong> The onChange event occurs when the form first appears. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectedIndex</td>
<td>Optional; Flash only</td>
<td>Used only for accordion and tabnavigator types; ignored otherwise. Specifies the page control to display as open, where 0 (not 1) specifies the first page control defined in the group.</td>
</tr>
<tr>
<td>startrow</td>
<td>Optional; Flash</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used only for the <code>repeater</code> type; ignored otherwise. Specifies the row number of the first row of the query to use in the Flash form repeater. This attribute is zero-based: the first row is row 0, not row 1 (as in most ColdFusion tags).</td>
</tr>
<tr>
<td>style</td>
<td>Optional; Flash and XML</td>
<td>Flash: a Flash style specification in CSS format. For detailed information on specifying Flash styles, see Creating Forms in Flash in the Developing ColdFusion Applications. XML: an inline CSS style specification.</td>
</tr>
<tr>
<td>tooltip</td>
<td>Optional; Flash</td>
<td>Text to display when the mouse pointer hovers in the form group area. If a control in the form group also specifies a tooltip, Flash displays the control's tooltip when the mouse pointer hovers over the control.</td>
</tr>
<tr>
<td>visible</td>
<td>Optional; Flash</td>
<td><code>yes</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Boolean value specifying whether the controls in the form group are visible. If the controls are invisible, the space that would be occupied by visible controls is blank.</td>
</tr>
</tbody>
</table>
### Usage

This tag requires an end tag. This tag is ignored if the `cfform` type is HTML; any tag body's contents are interpreted as if the surrounding `cfformgroup` does not exist.

In Flash format forms, this tag organizes the contents of the form. It groups and arranges child tags. The body of this tag can contain the following tags; all other tags and text are ignored:

- `cfFormItem`
- `cfCalendar`
- `cfGrid`
- `cfInput`
- `cfSelect`
- `cftextarea`
- `cftree`

For more information on using this tag in Flash forms, see [Creating Forms in Flash](#) in the *Developing ColdFusion Applications*. In XML format, ColdFusion passes the tag and its attributes to the XML; it is the responsibility of the skin XSLT to handle the XML. The ColdFusion basic skin supports the horizontal, vertical, and dualselect list styles only. For more information on using this tag in XML forms, see [Creating Forms in Flash](#) in the *Developing ColdFusion Applications*.

#### Example

For a simple example of an XML form that uses a single `cfformgroup` tag, see `cfform`. The following example shows how to use the `cfformgroup` tag to arrange elements on a Flash form. It creates an `hdividedbox` container that has a `vbox` container on each side. The left box has heading text and two radio buttons. The right box has heading text and three check boxes.

| width | Optional; Flash and XML | Width of the group container, in pixels. If you omit this attribute, Flash automatically sizes the container width. Ignored for Flash repeater type. |
<h3>Simple cfformgroup Example</h3>
<cfform name="myform" height="450" width="500" format="Flash">
  <cfformgroup type="hdividedbox">
    <cfformgroup type="VBox">
      <cfformitem type="text" height="20">
        Pets:
      </cfformitem>
      <cfinput type="Radio" name="pets" label="Dogs" value="Dogs" checked>
      <cfinput type="Radio" name="pets" label="Cats" value="Cats">
    </cfformgroup>
    <cfformgroup type="VBox">
      <cfformitem type="text" height="20">
        Fruits:
      </cfformitem>
      <cfinput type = "Checkbox" name="chk1" Label="Apples" value="Apples">
      <cfinput type="Checkbox" name="chk2" Label="Bananas" value="Bananas">
      <cfinput type="Checkbox" name="chk3" Label="Pears" value="Pears">
    </cfformgroup>
  </cfformgroup>
</cfform>

The following more complex example shows more fully how you can use cfformgroup tags to arrange controls in a Flash form. It also shows many of the text formatting features that you can use in a text cfformgroup body.

When you submit the form, the page dumps the contents of the Forms scope, to show you the submitted data.

<h2>cfformgroup Example</h2>
<cfif IsDefined("form.oncethrough")>
  <h3>The form submitted the following information to ColdFusion:</h3>
  <cfdump var="#form#"><br><br><br>
</cfif>

<h3>A Flash form using cfformgroup tags</h3>
<cfform name="myform" height="450" width="500" format="Flash">
  <!--- The following formgroup shows how you can present formatted text. --->
  <cfformitem type="html">
    <b><font color="#FF0000" size="+4" face="serif">This form has two tabs, asking for the following:<br>
    <li>contact information</li><br>
    <li>preferences</li><br>
    Try entering information on both tabs<br>Submit the form and see what ColdFusion gets in the Forms scope.<a href="http://www..com/" target="_blank">
    <font color="#0000FF">This link displays the home page in a new browser window</font></a><br>
    &nbsp;<br>
  </cfformitem>

  <!--- Use a tabnavigator with two tabs for user input. --->
  <cfformgroup type="tabnavigator" height="220">
    <cfformgroup type="page" label="Contact Information">
      <cfformitem type="text" height="20">
        Pets:
      </cfformitem>
      <cfinput type="Radio" name="pets" label="Dogs" value="Dogs" checked>
      <cfinput type="Radio" name="pets" label="Cats" value="Cats">
    </cfformgroup>
    <cfformgroup type="page" label="Preferences">
      <cfformitem type="text" height="20">
        Fruits:
      </cfformitem>
      <cfinput type = "Checkbox" name="chk1" Label="Apples" value="Apples">
      <cfinput type="Checkbox" name="chk2" Label="Bananas" value="Bananas">
      <cfinput type="Checkbox" name="chk3" Label="Pears" value="Pears">
    </cfformgroup>
  </cfformgroup>
</cfform>
<!---- Align the first and last name fields horizontally ---->
<cfformgroup type="horizontal" label="Your Name">
  <cfinput type="text" required="Yes" name="firstName" label="First" value="" width="100"/>
  <cfinput type="text" required="Yes" name="lastName" label="Last" value="" width="100"/>
</cfformgroup>
<cfformitem type="html"><textformat indent="95"><font size="-2">
Flash fills the email field in automatically.
You can replace any of the text.
</font></textformat>
</cfformitem>
<!---- The bind attribute gets the field contents from the firstname and lastName fields as they get filled in. ---->
<cfinput type="text" name="email" label="email" bind="{firstName.text}.{lastName.text}@mm.com">
<cfinput type="text" name="phone" validate="telephone" required="Yes" label="Phone Number">
</cfformgroup>
<cfformgroup type="page" label="Preferences">
  <cfformitem type="text" height="30">
    <b>Tell us your preferences</b>
  </cfformitem>
  <!---- Put the pet selectors to the left of the fruit selectors. ---->
  <cfformgroup type="hbox">
    <!---- Group the pet selector box contents, aligned vertically. ---->
    <cfformgroup type="vbox">
      <cfformitem type="text" height="20">
        Pets:
      </cfformitem>
      <cfformgroup type="vertical">
        <cfinput type="Radio" name="pets" label="Dogs" value="Dogs" checked>
        <cfinput type="Radio" name="pets" label="Cats" value="Cats">
      </cfformgroup>
    </cfformgroup>
  </cfformgroup>
  <!---- Group the fruit selector box contents, aligned vertically. ---->
  <cfformgroup type="vbox">
    <cfformitem type="text" height="20">
      Fruits:
    </cfformitem>
    <cfformgroup type="tile" width="200" label="Tile box">
      <!-- Flash requires unique names for all controls ---->
      <cfinput type = "Checkbox" name="chk1" Label="Apples" value="Apples">
      <cfinput type="Checkbox" name="chk2" Label="Bananas" value="Bananas">
      <cfinput type="Checkbox" name="chk3" Label="Pears" value="Pears">
      <cfinput type="Checkbox" name="chk4" Label="Oranges" value="Oranges">
      <cfinput type="Checkbox" name="chk5" Label="Grapes" value="Grapes">
      <cfinput type="Checkbox" name="chk6" Label="Cumquats" value="Cumquats">
    </cfformgroup>
  </cfformgroup>
</cfformgroup>
</cfformgroup>
</cfformgroup>
</cfformgroup>

<cfformgroup type="horizontal">
<cfinput type = "submit" name="submit" width="100" value = "Show Results">
<cfinput type = "reset" name="reset" width="100" value = "Reset Fields">
<cfinput type = "hidden" name="oncethrough" value = "Yes"
</cfformgroup>
</cfform>
**cfformitem**

**Description**

Inserts a horizontal line, a vertical line, a spacer, or text in a Flash form. Used in the `cfform` or `cfformgroup` tag body for Flash and XML forms. Ignored in HTML forms.

**Category**

Forms tags

**Syntax**

```xml
<cfformitem
  type = "hrule|vrule|spacer"
  height = "pixels"
  style = "style specification"
  visible = "yes|no"
  width = "pixels"/>

OR

<cfformitem
  type = "html|text|script"
  bind = "bind expression"
  enabled = "yes|no"
  height = "pixels"
  style = "style specification"
  tooltip = "text"
  visible = "yes|no"
  width = "pixels">
    ...text </cfformitem>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfapplet, cfform, cfformgroup, cfgrid, cfinput, cfselect, cfslider, cftextarea, cftree. Adding text, images, rules, and space with the cfformitem tag in Building Flash forms in the Developing ColdFusion Applications

**History**

ColdFusion MX 7.01: Added the "script" value for type attribute.

ColdFusion MX 7: Added tag

**Attributes**

The following table lists the attributes and their behavior in Flash forms. For XML format, if not otherwise noted, the
attribute is passed to the XML but is not interpreted by the basic XSL style sheet provided with ColdFusion.

⚠️ **Note**

Attributes that are marked as Flash only are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML in all controls except text and html types.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt; formats</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Required;Flash and XML</td>
<td>Flash:</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>html</td>
<td>place the text in the body of this tag on the form. For Flash forms, you can use the following text formatting tags, most of which correspond to HTML tags, in the text: a, b, br, font, i, img, li, p, textformat, and u. For details on using these formatting tags, see the Flash documentation. The style attribute has no effect on the format of the text in type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>place the text in the body of this tag on the form verbatim, without interpreting any markup. You can control the overall appearance of the text by using the style attribute.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spacer</td>
<td>places an invisible spacer of the specified height and width on the form. Used to place space between form controls. This tag must not have any children.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hrule</td>
<td>places a horizontal rule on the form. This tag must not have any children.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vrule</td>
<td>places a vertical rule on the form. This tag must not have any children.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>script</td>
<td>lets you create functions in Flash forms, which reduces the possibility of reaching the 64 KB limit.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>XML:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <strong>html:</strong> puts the CFML tag's body text in a CDATA section in an XML <code>xf:output</code> element.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <strong>text:</strong> XML-formats (escapes characters such as <code>&lt;</code>) the CFML tag's body text and puts it in a CDATA section in an XML <code>xf:output</code> element.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <strong>hrule:</strong> puts an <code>hr</code> tag in the output. Use the <code>style</code> attribute to specify all rule characteristics, including height and width. This tag must not have any children.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Any other string: generates an XML <code>xf:group</code> element with the type name as the <code>appearance</code> attribute. The CFML tag body is put in a CDATA section in a <code>cf:attribute</code> name=&quot;body&quot; element. The XSL transforms provided with ColdFusion ignore these elements.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>bind</strong></td>
<td>Optional;Flash</td>
<td>A Flash bind expression that populates the field with information from other form fields. If you use this attribute, ColdFusion ignores any text that you specify in the body of the <code>cftextitem</code> tag. This attribute can be useful if the <code>cformitem</code> tag is in a <code>cformgroup</code> tag. For more information, see Flash form data binding in the <code>cfinput</code> tag description.</td>
<td></td>
</tr>
<tr>
<td><strong>enabled</strong></td>
<td>Optional;Flash</td>
<td><strong>yes</strong></td>
<td>Boolean value that specifies whether the control is enabled. Disabled text appear in light gray. Has no effect on spacers and rules.</td>
</tr>
<tr>
<td><strong>height</strong></td>
<td>Optional;Flash</td>
<td></td>
<td>Height of the item, in pixels. If you omit this attribute, Flash automatically sizes the height. In ColdFusion XSL skins, use the <code>style</code> attribute, instead.</td>
</tr>
</tbody>
</table>
| **style** | Optional; Flash and XML | **Flash:**  
| --- | --- | ---  
|  |  | • Must be a style specification in CSS format.  
|  |  | Ignored if the `type` attribute is `html`.  
|  |  | For detailed information on specifying Flash styles, see [Creating Forms in Flash](link) in the *Developing ColdFusion Applications*. Not used with the spacer type.  
|  |  | **XML:**  
|  |  | • ColdFusion passes the `style` attribute to the XML. ColdFusion skins include the `style` attribute in the generated HTML.  

| **tooltip** | Optional; Flash | **tooltip**  
| --- | --- | ---  
|  | Text to display when the mouse pointer hovers over the control. Has no effect on spacers.  

| **visible** | Optional; Flash | **yes**  
| --- | --- | ---  
|  | Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank. Has no effect on spacers.  

| **width** | Optional; Flash | **width**  
| --- | --- | ---  
|  | Width of the item, in pixels. If you omit this attribute, Flash automatically sizes the width. In ColdFusion XSL skins, use the `style` attribute, instead.  

### Usage

This tag requires an end tag or a slash before the closing end character of the opening tag, as the following example shows:

```xml
<cfformitem type="hrule" />
```
For more information on using this tag in Flash forms, see Creating Forms in Flash in the Developing ColdFusion Applications.

Example

The following example shows a simple Flash form by using horizontal rules and text:

```xml
<h3>cfformitem Example</h3>
<cfform name="myform" height="450" width="500" format="Flash">
  <cfformitem type="hrule" />
  <cfformitem type="text">
    This simple form has two hrule cfformitem tags around the cfformitem tag that contains this text.
  </cfformitem>
  <cfformitem type="hrule" />
</cfform>
```

For a more complex form, see cfformgroup.
**cfftp**

**Description**

Lets users implement File Transfer Protocol (FTP) operations.

**Category**

File management tags, Internet protocol tags

**Syntax**

The tag syntax depends on the `action` attribute value. See the following sections:

- cfftp: Opening and closing FTP server connections
- cfftp: Opening and closing secure FTP server connections
- cfftp: Connection: file and directory operations
- `cfftp action = "listDir"`

**See also**

cfhttp, cfdap, cfmail, cfpop; Performing file operations with cfftp in Interacting with Remote Servers in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added the fingerprint, key, paraphrase, and secure attributes to support secure FTP. Added the values ="quote", "site", "allo", and "acct" to the `action` attribute. ColdFusion MX 7: Added the `result` attribute for file and directory operations. ColdFusion MX: Deprecated the `agentname` attribute. It might not work, and might cause an error, in later releases.

**Usage**

Use this tag to move files between a ColdFusion server and an FTP server. This tag does not move files between a ColdFusion server and a client browser. You do this as follows:

- To transfer files from a client to a ColdFusion server: `cffile action = "upload"
- To transfer files from a ColdFusion server to a client: the `cfcontent` tag

**Security settings**

ColdFusion security settings can prevent the `cfftp` tag from executing. If you run ColdFusion applications on a server that is used by multiple customers, consider the security of the files that the customer can move. For more information, see the Administering ColdFusion Security section of Configuring and Administering ColdFusion.
cftp: Opening and closing FTP server connections

Description

To establish a connection with an FTP server, use the open action with a connection attribute.

Syntax

```xml
<cfftp
    action = "open|close|quote|site|allo|acct"
    actionparam = "command or account information"
    buffersize = "number"
    connection = "name"
    passive = "yes|no"
    password = "password"
    port = "port"
    proxyServer = "proxy server"
    retryCount = "number"
    server = "server"
    stopOnError = "yes|no"
    timeout = "time-out in seconds"
    username = "name">
</cfftp>
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfhttp, cfldap, cfmail, cfpop

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>action</td>
<td>Required</td>
<td>FTP operation to perform.</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>---------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• open: creates an FTP connection.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• close: terminates an FTP connection.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• quote: sends a command verbatim to the FTP server.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• site: executes a site-specific command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• allo: allocates memory for operations, such as putting large files, on the server.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• acct: sends account information on systems that require it.</td>
<td></td>
</tr>
</tbody>
</table>

| actionparam | Optional | Used only when action is quote, site, or acct. Specifies the command when action is quote or site; specifies account information when action is acct. |

| buffersize | Optional | Buffer size in bytes. |

| connection | Optional, but always used with open or close | Name of the FTP connection. If you specify the username, password, and server attributes, and if no connection exists for them, ColdFusion creates one. Calls to cfftp with the same connection name reuse the connection. |

| passive | Optional | no |
|         |          | • yes: enables passive mode. |
|         |          | • no |

| password | Required if action = "open" | Password to log in the user. |

<p>| port | Optional | 21 | Remote port to which to connect. |</p>
<table>
<thead>
<tr>
<th><strong>proxyServer</strong></th>
<th>Optional</th>
<th>String. Name of proxy server (or servers) to use, if proxy access is specified.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>retryCount</strong></td>
<td>Optional</td>
<td>1 Number of retries until failure is reported.</td>
</tr>
<tr>
<td><strong>server</strong></td>
<td>Required if action = &quot;open&quot;</td>
<td>FTP server to which to connect; for example, ftp.myserver.com.</td>
</tr>
</tbody>
</table>
| **stopOnError** | Optional | yes • yes: halts processing, displays an appropriate error. 
• no: if secure="no", populates these variables:  
  • cfftp.succeeded: yes or no.  
  • cfftp.errorText: Message text. For conditional operations, use cfftp.errorCode. Do not use cfftp.errorText for this purpose. |
| **timeout**     | Optional | 30 Value in seconds for the time-out of all operations, including individual data request operations. |
| **username**    | Required if action = "open" | User name to pass in the FTP operation. |

**Usage**

When you establish a connection with cfftp action="open" and specify a name in the connection attribute, ColdFusion caches the connection so that you can reuse it to perform additional FTP operations. When you use a cached connection for subsequent FTP operations, you do not have to specify the username, password, or server connection attributes. The FTP operations that use the same connection name automatically use the information stored in the cached connection. Using a cached connection helps save connection time and improves...
file transfer performance. You do not need to open a connection for single, simple, FTP operations, such as GetFile or PutFile.

With any action except close, you can set the internal buffer size by specifying buffersize. If you specify quote, site, allo, or acct as the action and set secure="yes" an error is generated. You specify the command to send to the FTP server in the actionparam attribute when you specify site or quote as the action. When site is the action, you use the actionparam attribute to specify the site-specific information.

To keep a connection open throughout a session or longer, put the connection name in the Session or Application scope; for example, specify connection="Session.FTPConnection". However, if you do this, you must specify the full variable name in all FTP operations, and you must use the close action when you are finished. Keeping a connection open prevents others from using the FTP server; so close a connection as soon as possible. If you do not assign the connection name to Session or Application variable, the connection remains open for the current page only, and you do not have to close it manually. Changes to a cached connection, such as changing retryCount or timeout values, might require reestablishing the connection.

Example

```
<p>cfftp lets users implement File Transfer Protocol operations. By default, cfftp caches an open connection to an FTP server.</p>
<p>cfftp operations are usually of two types:</p>
<ul>
<li>Establishing a connection</li>
<li>Performing file and directory operations</li>
</ul>
<p>This example opens and verifies a connection, lists the files in a directory, and closes the connection.</p>
<p>Open a connection</p>
<cfftp action = "open" username = "anonymous" connection = "My_query" password = "youremail@email.com" server = "ftp.tucows.com" stopOnError = "Yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput></p>
<p>List the files in a directory:</p>
<cfftp action = "LISTDIR" stopOnError = "Yes" name = "ListFiles" directory = "/" connection = "my_query">
<cfoutput query = "ListFiles">
#name#<br>
</cfoutput>
</cfftp>
<p>Close the connection:</p>
<cfftp action = "close" connection = "My_query" stopOnError = "Yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput></cfoutput>
```
cfftp: Opening and closing secure FTP server connections

Description

To establish a connection with a secure FTP server, use the *open* action with a *connection* attribute, specify that *secure = "yes"*, and specify the key, passphrase, and fingerprint as appropriate.

```cfftp
action = "open|close"
connection = "name"
fingerprint = "ssh-dss.ssh-rsa"
key = "private key"
passive = "yes|no">
passphrase = "passphrase"
password = "password"
port = "port"
proxyServer = "proxy server"
retryCount = "number"
secure = "yes|no"
server = "server"
stopOnError = "yes|no"
timeout = "time-out in seconds"
username = "name">
```

⚠️ Note

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

See also

`cfhttp`, `cfldap`, `cfmail`, `cfpop`

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| action    | Required|         | FTP operation to perform.  
|           |         |         | • *open*: creates an FTP connection.  
<p>|           |         |         | • <em>close</em>: terminates an FTP connection. |</p>
<table>
<thead>
<tr>
<th><strong>connection</strong></th>
<th>Optional, but always used with open or close</th>
<th>Name of the FTP connection. If you specify the username, password, and server attributes, and if no connection exists for them, ColdFusion creates one. Calls to cfftp with the same connection name reuse the connection.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fingerprint</strong></td>
<td>Optional. Used only when server, username, and password are supplied</td>
<td>Fingerprint of the host key in the form ssh-dss.ssh-rsa, which is a 16-byte unique identifier for the server attribute that you specify. The fingerprint consists of eight pairs of hexadecimal values in the form hh:hh:hh:hh:hh:hh:hh:hh. ColdFusion checks the fingerprint of the remote server only if the fingerprint value is specified.</td>
</tr>
<tr>
<td><strong>key</strong></td>
<td>Required if action=&quot;open&quot; (}When (secure=&quot;yes&quot;, either password or key is required.)</td>
<td>Public-key based authentication. Refers to the absolute path to the private key of the user. Possession of a private key provides authentication by sending a signature created with a private key. The server must ensure that the key is a valid authentication for the user and that the signature is valid. Both must be valid to accept the authentication.</td>
</tr>
</tbody>
</table>
| **passive**     | Optional | no | Valid only if secure="no".  

- yes: enables passive mode.  
- no |

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>passphrase</td>
<td>Optional. Used when key is specified</td>
<td>Because private keys are stored in an encrypted form on the client host, the user must supply a passphrase to enable generating the signature.</td>
</tr>
<tr>
<td>password</td>
<td>Required if action=&quot;open&quot; ( {{secure=&quot;yes&quot;, either password or key is required.})</td>
<td>Password to log in the user.</td>
</tr>
<tr>
<td>port</td>
<td>Optional</td>
<td>Remote port to which to connect.</td>
</tr>
<tr>
<td>proxyServer</td>
<td>Optional</td>
<td>String. Name of proxy server (or servers) to use, if proxy access is specified.</td>
</tr>
<tr>
<td>retryCount</td>
<td>Optional</td>
<td>Number of retries until failure is reported.</td>
</tr>
</tbody>
</table>
| secure     | Optional                                                                    | • yes: enables secure FTP  
• no                                                                                  |
| server     | Required if action="open"                                                   | FTP server to which to connect; for example, ftp.myserver.com.       |
### `stopOnError`

- **Type:** Optional
- **Default:** `no`

- **yes:** halts processing, displays an appropriate error.
- **no:** If `secure="yes"`, populates the following variables:
  - `cfftp.succeeded`: `yes` or `no`
  - `cfftp.errorCode`: error number
  - `cfftp.errorText`: message text

- For all file operations, returns the following error codes:
  - `SSH-CONNECT 25`
  - `SSH_MSG_USERAUTH_FAILURE 51`
  - `SSH_MSG_USERAUTH_SUCCESS 52`
  - `SSH_MSG_REQUEST_SUCCESS 81`
  - `SSH_MSG_REQUEST_FAILURE 82`

For conditional operations, use `cfftp.errorCode`. Do not use `cfftp.errorText` for this purpose.

### `timeout`

- **Type:** Optional
- **Default:** `30`

Value in seconds for the time-out of all operations, including individual data request operations.

### `username`

- **Type:** Required if `action="open"`

User name to pass in the FTP operation.

---

**Usage**

The `cfftp` tag lets you open a connection to a Secure Shell (SSH) server by using either symmetric or asymmetric encryption. To use symmetric encryption, you specify `secure="yes"`, the user name, password, connection, and fingerprint. To use asymmetric encryption, first generate private-public key pairs for each user authorized to have access to the server. Each authorized user's public key is stored on the server; each user's private key is encrypted and stored on that user's computer. To open a connection to the SSH server, you specify `secure="yes"`, the user
name, the password, or the private key and the passphrase that the server uses to decrypt the private key, connection, and fingerprint. After you open the connection to the SSH server, you can use that connection for any action supported by the cfftp tag. To keep a connection open throughout a session or longer, put the connection name in the Session or Application scope; for example, specify connection="Session.FTPConnection". However, if you do this, specify the full variable name in all FTP operations, and use the close action when you are finished. Keeping a connection open prevents others from using the FTP server; so close a connection as soon as possible. If you do not assign the connection name to Session or Application variable, the connection remains open for the current page only, and you do not have to close it manually. Changes to a cached connection, such as changing retryCount or timeout values, might require reestablishing the connection.

Example

<!--- This example uses symmetric encryption. --->

<!--- Open the secure connection. --->
<cfftp action = "open"
    username = "myusername"
    connection = "My_query"
    password = "mypassword"
    server = "ftp.tucows.com"
    secure = "yes">
    <p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
    <cfdump var ="#My_query# label="connection">

<!--- Transfer files to the remote server. --->
<cfset absolutePathToLocalFile="C:\one\two\myfile.htm">
<cfif FileExists(absolutePathToLocalFile)>
    <cfftp action = "putFile"
        connection="My_query"
        localFile="#variables.absolutePathToLocalFile#"
        remoteFile="/home/myname/sftptest/myfile.htm">
<cfelse>
    <!--- Put error handling code here. --->
    </cfif>
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>

<!--- Close the connection. --->
<cfftp action="close" connection="My_query">
<!--- This example uses asymmetric encryption. --->

<!--- Open the secure connection. --->
<cfftp action = "open"
    username = "myusername"
    connection = "My_query"
    key="C:\mykeys\myprivatekey"
    passphrase = "zHx628Fg"
    server = "ftp.tucows.com"
    secure = "yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
<cfdump var="#My_query# label="connection">

<!--- List files on the remote server. --->
<cftry>
<!--- List the files in a directory. --->
<cfftp action = "listDir"
    connection="My_query"
    stopOnError="yes"
    name="ListFiles"
    directory="/">
<cfcatch>
<!--- Close the connection. --->
<cfftp action="close" connection="My_query" stopOnError="no">
</cfcatch>
</cftry>
cftp: Connection: file and directory operations

Description

To perform file and directory operations with cftp, use this form of the cftp tag.

Syntax

```
<cftp
    action = "action"
    ASCIIExtensionList = "extensions"
    connection = "connection name"
    directory = "directory name"
    existing = "file or directory name"
    failIfExists = "yes|no"
    item = "directory or file"
    localFile = "filename"
    name = "query name"
    new = "file or directory name"
    passive = "yes|no"
    password = "password"
    proxyServer = "proxy server"
    remoteFile = "filename"
    result = "result name"
    server = "server"
    timeout = "time-out in seconds"
    transferMode = "ASCII FTP|Binary FTP|Auto FTP"
    username = "name">
```

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfhttp, cfldap, cfmail, cfpop

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
| action       | Required if connection is not cached | FTP operation to perform:  
|             |                                    | • changedir  
|             |                                    | • createDir  
|             |                                    | • listDir  
|             |                                    | • removeDir  
|             |                                    | • getFile  
|             |                                    | • putFile  
|             |                                    | • rename  
|             |                                    | • remove  
|             |                                    | • getCurrentDir  
|             |                                    | • getCurrentURL  
|             |                                    | • existsDir  
|             |                                    | • existsFile  
|             |                                    | • exists  
| ASCIIExtensionList | Optional | Delimited list of file extensions that force ASCII transfer mode, if transferMode = "auto".  
| connection   | Required if action = "open" or "closed" | The name of the FTP connection. Used to cache a new FTP connection or to reuse an existing connection.  
| directory    | Required if action = "changedir", "createDir", "listDir", or "existsDir" | Directory on which to perform an operation.  
| existing     | Required if action = "rename" | Current name of the file or directory on the remote server.  
| failIfExists | Optional | yes  
| item         | Required if action = "exists" or "remove" | • yes: If a local file with same name exists, the getFile action fails.  
|             |                                    | • no  
| localFile    | Required if action = "getFile" or "putFile" | Name of the file on the local file system. For more information, see Usage.  
| name         | Required if action = "listDir" | Query name of directory listing.  

<table>
<thead>
<tr>
<th>new</th>
<th>Required if action = &quot;rename&quot;</th>
<th>New name of file or directory on the remote server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>passive</td>
<td>Optional</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: enables passive mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no</td>
</tr>
<tr>
<td>password</td>
<td>Required if action = &quot;open&quot;</td>
<td>Password to log in the user.</td>
</tr>
<tr>
<td>proxyServer</td>
<td>Optional</td>
<td>String. Name of the proxy servers to use, if proxy access is specified.</td>
</tr>
<tr>
<td>remoteFile</td>
<td>Required if action = &quot;getFile&quot;, &quot;putFile&quot;, or &quot;existsFile&quot;</td>
<td>Name of the file on the FTP server file system.</td>
</tr>
<tr>
<td>result</td>
<td>Optional</td>
<td>Specifies a name for the structure in which cfftp stores the returnValue variable. If set, this value replaces cfftp as the prefix to use when accessing returnVariable. For more information, see Usage.</td>
</tr>
<tr>
<td>server</td>
<td>Required if FTP connection is not cached</td>
<td>FTP server to which to connect; for example, ftp.myserver.com.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional</td>
<td>30 seconds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The length of time, in seconds, that ColdFusion waits for a response from the FTP server. Used with action = &quot;open&quot; for a cached connection.</td>
</tr>
<tr>
<td>transferMode</td>
<td>Optional</td>
<td>auto</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ASCII FTP transfer mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Binary FTP transfer mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Auto FTP transfer mode</td>
</tr>
<tr>
<td>username</td>
<td>Required if connection is not cached</td>
<td>User name to pass in the FTP operation.</td>
</tr>
</tbody>
</table>

Usage
If you use connection caching to an active FTP connection, you do not have to respecify the username, password, or server connection attributes. Changing a cached connection, such as changing retryCount or timeout values, might require reestablishing the connection. If action = "listDir", the attributes column returns directory or normal. Other platform-specific values, such as hidden and system, are no longer supported. If action = "listDir", a mode column is returned. The column contains an octal string representation of UNIX permissions; for example, "777." The cfftp.returnValue variable provides the return value for these actions:

- getCurrentDir
- getCurrentURL
- existsDir
- existsFile
- exists

For more information, see the Developing ColdFusion Applications.

For more information, see the section Performing file operations with cfftp in Developing ColdFusion Applications.

**localFile attribute**

Use the following syntax to specify an in-memory file, which is not written to disk, as the local file. In-memory files speed processing of transient data.

```
ram:///filepath
```

The filepath can include directories, for example ram:///petStore/images/poodle.jpg. Create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications.

**Action (cfftp.ReturnValue variable)**

The results of an action determine the value of the returnValue variable, as the following table shows:

<table>
<thead>
<tr>
<th>cfftp action</th>
<th>Value of cfftp.returnValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>getCurrentDir</td>
<td>String. Current directory.</td>
</tr>
<tr>
<td>getCurrentURL</td>
<td>String. Current URL.</td>
</tr>
<tr>
<td>existsDir</td>
<td>yes or no.</td>
</tr>
<tr>
<td>existsFile</td>
<td>yes or no.</td>
</tr>
<tr>
<td>exists</td>
<td>yes or no.</td>
</tr>
</tbody>
</table>

To access the returnValue variable, you must prefix it with either cfftp or the value specified by the result attribute, if it is set. The result attribute provides a way for cfftp calls from multiple pages, possibly at the same time, to avoid overwriting the results of one with another. If you set the result attribute to myResult, for example, you would access the returnVariable variable as myResult.returnVariable. Otherwise, you would access it as cfftp.returnVariable.
Example

The following example opens a connection and gets a file that lists file or directory name, path, URL, length, and modification date:

```cftplang
<p>Open a connection
<cfftp connection = "myConnection"
  username = "myUserName"
  password = "myUserName@allaire.com"
  server = "ftp.allaire.com"
  action = "open"
  stopOnError = "Yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
<cfftp connection = "myConnection"
  action = "LISTDIR"
  stopOnError = "Yes"
  name = "ListDirs"
  directory = "/">
</cfftp>

<p>FTP Directory Listing:<br>
<cftable query = "ListDirs" HTMLTable = "Yes" colHeaders = "Yes">
  <cfcol header = "<b>Name</b>" text = "#name#">
  <cfcol header = "<b>Path</b>" text = "#path#">
  <cfcol header = "<b>URL</b>" text = "#url#">
  <cfcol header = "<b>Length</b>" text = "#length#">
  <cfcol header = "<b>LastModified</b>" text = "#DateFormat(lastmodified)#"/>
  <cfcol header = "<b>IsDirectory</b>" text = "#isdirectory#">
</cftable>
</p>

<p>Move Image File to Remote Server:<br></p>
<!--- The image will be put into the root directory of the FTP server unless otherwise noted, i.e., remoteFile = "somewhere_put.jpg" vs remoteFile = "/support/somewhere_put.jpg" --->
<cfftp
  connection = "myConnection"
  action = "putFile"
  name = "uploadFile"
  transferMode = "binary"
  localFile = "C:\files\upload\somewhere.jpg"
  remoteFile = "somewhere_put.jpg">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
</cfftp>

<p>Close the connection:
<cfftp connection = "myConnection"
  action = "close"
  stopOnError = "Yes">
<p>Did it succeed? <cfoutput>#cfftp.succeeded#</cfoutput>
</cfftp>
```
cfftp action = "listDir"

Description

To access the columns in a query object, use this tag with action = "listDir".

Usage

When you use this action, specify a value for the name attribute. This value holds the results of the listDir action in a query object. The query object consists of columns that you can reference, in the form queryname.columnname, where queryname is the name of the query, specified in the name attribute; and columnname is a column returned in the query object. The value row is the row number of each file/directory entry returned by the listDir operation. A separate row is created for each entry:

<table>
<thead>
<tr>
<th>cfftp query object column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Filename of the current element.</td>
</tr>
<tr>
<td>Path</td>
<td>File path (without drive designation) of the current element.</td>
</tr>
<tr>
<td>URL</td>
<td>Complete URL for the current element (file or directory).</td>
</tr>
<tr>
<td>Length</td>
<td>File size of the current element.</td>
</tr>
<tr>
<td>LastModified</td>
<td>Unformatted date/time value of the current element.</td>
</tr>
<tr>
<td>Attributes</td>
<td>String. Attributes of the current element: normal or Directory.</td>
</tr>
<tr>
<td>IsDirectory</td>
<td>Boolean. Whether object is a file or directory.</td>
</tr>
<tr>
<td>Mode</td>
<td>Applies only to UNIX and Linux. Permissions. Octal string.</td>
</tr>
</tbody>
</table>

⚠️ Note

Previously supported query column values that pertain to system-specific information are not supported; for example, hidden and system.
cform

Description

Builds a form with CFML custom control tags; these provide more functionality than standard HTML form input elements. You can include the resulting form on the client page as HTML or Adobe Flash content, and generate the form by using XML and XSLT.

Category

Forms tags

Syntax

```cfml
<cfform
   accessible = "yes|no"
   action = "form action"
   archive = "URL"
   codeBase = "URL"
   format = "HTML|Flash|XML"
   height = "pixels|percent"
   id = "HTML id"  method = "POST|GET"
   name = "name"
   onError = "JavaScript function name or ActionScript code"
   onLoad = "load event script"  onReset = "reset event script"  onSubmit = "JavaScript"
   onSuccess = "JavaScript function name"
   preloader = "yes|no"
   preserveData = "yes|no"
   scriptSrc = "path"
   skin = "Flash skin|XSL skin"
   style = "style specification"
   timeout = "seconds"
   width = "pixels|percent"
   wMode = "window|transparent|opaque">
   ...
</cfform>
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfajaximport, cfapplet, cfcalendar, cfformgroup, cfformitem, cfgrid, cfinput, cfselect, cfslid

er, cftextarea, cftree; Requesting and Presenting Information in the Developing ColdFusion Applications

History

ColdFusion 8:
• Added support for adding interactive fields in PDF forms.
• Added the onSuccess attribute and support in AJAX controls for the onError attribute

ColdFusion MX 7:
• Added ability to set the default value of the scriptSrc attribute in the ColdFusion Administrator.
• Deprecated the passthrough attribute. The tag now supports all HTML form tag attributes directly.
• Added the method attribute and support for the GET method.
• Added support for Flash and XML output, including the format, height, width, preloader, timeout, wmode, accessible, and skin attributes.
• Added cfformgroup, cfformitem, and cftextarea child tags.
• Added the onReset attribute.

ColdFusion MX:
• Deprecated the enableCAB attribute. It might not work, and might cause an error, in later releases.
• Changed the name and action attributes to optional.
• Changed integer validation to require an integer value. In previous releases it would convert a floating point value to an integer.

Attributes

The following table lists attributes that ColdFusion uses directly. For HTML format forms, this tag also supports the standard HTML form tag attributes that are not on this list, and passes them directly to the browser. ColdFusion also includes all supported HTML attributes in the XML.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Applies to</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessible</td>
<td>Flash</td>
<td>Opt</td>
<td>no</td>
<td>Specifies whether to include support screen readers in the Flash form.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Screen reader support adds approximately 80 KB to the SWF file sent to the client.</td>
</tr>
<tr>
<td>action</td>
<td>FlashHTMLXML</td>
<td>Opt</td>
<td>See Description</td>
<td>Name of ColdFusion page to execute when the form is submitted for processing. If you omit this attribute and the method is get, the form posts to the page identified by the CGI.SCRIPT_NAME variable (the requested page that resulted in displaying the form). If the method is post, the form posts to the page identified by the CGI.QUERY_STRING variables.</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>-----</td>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>archive</td>
<td>applets in HTML and XML</td>
<td>Opt</td>
<td>/CFIDE/classes/cfapplets.jar</td>
<td>URL of downloadable Java classes for cfgrid, cfslider, and cftree applet controls.</td>
</tr>
<tr>
<td>codeBase</td>
<td>applets in HTML and XML</td>
<td>Opt</td>
<td>/CFIDE/classes/cf-j2re-win.cab</td>
<td>URL of downloadable JRE plug-in for Internet Explorer; used for cfgrid, cfslider, and cftree Java applet controls.</td>
</tr>
<tr>
<td>format</td>
<td>FlashHTMLXML</td>
<td>Opt</td>
<td>HTML</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>-----</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• HTML: generates an HTML form and send it to the client. cfgrid and cftree child controls can be in Flash or applet format.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Flash: generates a Flash form and send it to the client. All controls are in Flash format.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• XML: generates XForms-compliant XML and save the results in a variable specified by the name attribute. By default, ColdFusion also applies an XSL skin and displays the result. For more information, see the skin attribute.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>height</th>
<th>FlashXML</th>
<th>Opt</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>The height of the form. Use a number to specify pixels. In Flash, you can use a percentage value, such as &quot;height=60%&quot; to specify a percentage of the available width. The displayed height might be less than the specified size. <strong>Note:</strong> The width and height attributes are required for Flash forms, if they are used inside of a table.</td>
</tr>
</tbody>
</table>

<p>| id     | name attribute value | the HTML id of the form. |</p>
<table>
<thead>
<tr>
<th>method</th>
<th>FlashHTML XML</th>
<th>Opt</th>
<th>post</th>
</tr>
</thead>
</table>
|        | The method the browser uses to send the form data to the server:  
|        | • **post**: sends the data using the HTTP post method. This method sends the data in a separate message to the server.  
|        | • **get**: sends the data using the HTTP get method, which puts the form field contents in the URL query string. |

<table>
<thead>
<tr>
<th>name</th>
<th>FlashHTMLXML</th>
<th>Opt</th>
<th>CFForm_n</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A name for the form. In HTML format, if you omit this attribute and specify an <code>id</code> attribute, ColdFusion does not include a name attribute in the HTML sent to the browser; this behavior lets you use the <code>cfform</code> tag to create XHTML-compliant forms. If you omit the <code>name</code> attribute and the <code>id</code> attribute, ColdFusion generates a name of the form <code>CFForm_n</code> where <code>n</code> is a number that is assigned serially to the forms on a page.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Format</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>onError</td>
<td>FlashHTML</td>
<td>Opt</td>
<td>For Flash format forms: Applies only for onSubmit or onBlur validation; has no effect for onServer validation. An ActionScript expression or expressions to execute if the user submits a form with one or more validation errors. For HTML format forms: Applies only to forms inside cfd iv, cflayout, cfp od, or cfwindow controls. The name of a JavaScript function that runs if an asynchronous form submission fails. For more information, see the Usage section.</td>
</tr>
<tr>
<td>onLoad</td>
<td>HTMLXML</td>
<td>Opt</td>
<td>JavaScript to execute when the form loads.</td>
</tr>
<tr>
<td>onReset</td>
<td>HTMLXML</td>
<td>Opt</td>
<td>JavaScript to execute when the user clicks a reset button.</td>
</tr>
<tr>
<td>onSubmit</td>
<td>FlashHTMLXML</td>
<td>Opt</td>
<td>JavaScript or ActionScript function to execute to preprocess data before form is submitted. If any child tags specify onSubmit field validation, ColdFusion does the validation before executing this JavaScript.</td>
</tr>
<tr>
<td>onSuccess</td>
<td>HTML</td>
<td>Opt</td>
<td>Applies only to forms inside <code>cfdiv</code>, <code>cflayout</code>, <code>cfpod</code>, or <code>cfwindow</code> controls. The name of a JavaScript function that runs when an asynchronous form submission succeeds. For more information see the Usage section.</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>-----</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>preloader</td>
<td>Flash</td>
<td>Opt</td>
<td>yes</td>
</tr>
<tr>
<td>preserveData</td>
<td>HTMLXML</td>
<td>Opt</td>
<td>no</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>-----</td>
<td>----</td>
</tr>
</tbody>
</table>

When the `cfform` action attribute posts back to the page that contains the form, this attribute determines whether to override the control values with the submitted values.

- **no**: uses values specified in the control tag attributes.
- **yes**: uses corresponding submitted values. Applies to these controls:
  - `cfinput`, `cfslider`, `cftextinput`: overrides the value attribute value.
  - `cfselect` controls that are populated from queries: overrides the `selected` attribute. See `cfselect`.
  - `cftree` controls: overrides the `cftreeitem expand` attribute. If `yes`, expands previously-selected elements. The `cftree complete Path` attribute must be set to `yes`.
  - `cfgrid` controls: has no effect. (This avoids confusion as to whether data has been resubmitted to the database by the control.)
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scriptSrc</td>
<td>FlashHTMLXML</td>
<td>Opt</td>
<td>Specifies the URL, relative to the web root, of the directory that contains ColdFusion JavaScript files, including the cfform.js file with the client-side JavaScript used by this tag and its child tags. For XML format forms, this directory is also the default directory for XSLT skins. When you use this attribute, the specified directory must have the same structure as the /CFIDE/scripts directory. For example, if you specify scriptsrc=&quot;/resources/myScripts&quot;, the JavaScript files used by ColdFusion AJAX features must be in the /resources/myScripts/ajax directory. This attribute is useful if the file is not in the default location. This attribute may be required in some hosting environments and configurations that block access to the /CFIDE directory. The location is set in the ColdFusion Administrator; by default, it is /CFIDE/scripts. <strong>Notes:</strong> If you specify this attribute, copy the CF_RunActiveConte...</td>
</tr>
</tbody>
</table>
nt.js file from the CFIDE/scripts directory to the specified directory. You can have only one scriptsrc attribute on a page, including any cfajaximport tag scriptsrc) attribute. If you have multiple cfform tags, you can specify the scriptsrc attribute in
FlashXML

<table>
<thead>
<tr>
<th>skin</th>
<th>FlashXML</th>
<th>Opt</th>
<th>Flash: {{\text{haloGreen}}}XML: default.xsl</th>
</tr>
</thead>
</table>

Flash: Use a halo color to stylize the output. The skin determines the color used for highlighted and selected elements.

- haloSilver
- haloBlue
- haloGreen
- haloOrange

XML: Specifies whether to apply an XSL skin and display the resulting HTML to the client. Can be any of the following:
- ColdFusion skin name: applies the specified skin.
- XSL file name: applies the skin located in the specified path.
- none: does not apply an XSL skin. Your CFML page must process the XML that ColdFusion saves in the variable specified by the name attribute, and display any results.
- omitted or default: uses the ColdFusion default skin. You can specify the following ColdFusion skins (located in the cf_webroot):
  - basic
<table>
<thead>
<tr>
<th>style</th>
<th>HTML, Flash, XML</th>
<th>Opt</th>
</tr>
</thead>
</table>

A filename can be any of the following:
- absolute URL
- URL relative to the web root
- absolute file path
- name of a file in the scripts folder or a subdirectory of the `cf_webroot\CFIDE\scripts` directory. In this case, do not specify the `.xsl` suffix.

Styles to apply to the form. In HTML or XML format, ColdFusion passes the style attribute to the browser or XML. In Flash format, must be a style specification in CSS format. For detailed information on specifying Flash styles, see Creating Forms in Flash in the Developing ColdFusion Applications.
<table>
<thead>
<tr>
<th><strong>timeout</strong></th>
<th>Flash</th>
<th>Opt</th>
<th>0</th>
<th>Integer number of seconds for which to keep the form data in the Flash cache on the server. A value of 0 prevents the data from being cached. For more information, see Caching data in Flash forms in Best practices for Flash forms in the Developing ColdFusion Applications.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>width</strong></td>
<td>FlashXML</td>
<td>Opt</td>
<td>100%</td>
<td>The width of the form. Use a number to specify pixels. In Flash, you can use a percentage value, such as &quot;width=60 %&quot; to specify a percentage of the available width. Note: The width and height attributes are required for Flash forms, if they are used inside of a table.</td>
</tr>
<tr>
<td>wMode</td>
<td>Flash</td>
<td>Opt</td>
<td>window</td>
<td>Specifies how the Flash form appears relative to other displayable content that occupies the same space on an HTML page.</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-----</td>
<td>---------</td>
<td>----------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>window</td>
<td>• <strong>window</strong>: the Flash form is the topmost layer on the page and obscures anything that would share the space, such as drop-down dynamic HTML lists.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>transparent</td>
<td>• <strong>transparent</strong>: the Flash form honors the z-index of dhtml so you can float items above it. If the Flash form is above any item, transparent regions in the form show the content that is below it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>opaque</td>
<td>• <strong>opaque</strong>: the Flash form honors the z-index of dhtml so you can float items above it. If the Flash form is above any item, it blocks any content that is below it.</td>
</tr>
</tbody>
</table>

⚠️ **Note**

Attributes that are not marked as supported in XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML as html namespace attributes to the `form` tag.

**Usage**

This tag requires an end tag.

You can use the following ColdFusion form control tags in the `cfform` tag:
• **cfapplet**: Used in HTML and XML format only; embeds a registered Java applet.
• **cfformgroup**: Used in Flash and XML format only; groups and arranges child controls.
• **cfformitem**: Used in Flash and XML format only; adds horizontal rules, vertical rules, and text to the form.
• **cfgrid**: Creates a grid control to display tabular data.
• **cfinput**: Creates and an input element.
• **cfselect**: Creates a drop-down list box.
• **cfslider**: Used in HTML and XML format only; creates a slider control.
• **cftextarea**: Creates a multiline text input box.
• **cftree**: Creates a tree control.

In HTML format, all tags, and in Flash format the cftree and cfgrid tags, require JavaScript support on the browser. The cfapplet tag and applet format cfgrid, cfslider, and cftree tags require the client to download a Java applet. If you specify Flash format in the cfform tag, ColdFusion ignores any HTML in the form body. Use ColdFusion tags, such as cfinput, for all form controls. You can include individual Flash format cfgrid and cftree controls in an HTML format cfform tag.

In Flash format, if your forms do not request sensitive data (such as credit card numbers), consider setting the **timeout** attribute. This can prevent users from getting "The form data has expired. Please reload this page in your browser" errors if they use the browser back button to return to the form. For more information, see *Caching data in Flash forms* in *Best practices for Flash forms* in the *Developing ColdFusion Applications*.

**Note**

In Flash format, if you do not specify **height** and **width** attributes, Flash reserves browser space equal to the area of the browser window. If any other output follows the form, users must scroll to see it. Therefore, if you follow a Flash form with additional output, specify the **height** and **width** values. The **width** and **height** attributes are required for Flash forms, if they are used inside of a table.

If attribute value text must include quotation marks, escape them by doubling them.

### Using the onError attribute in Flash forms

If you use onSubmit or onBlur validation, the **onError** attribute lets you specify ActionScript code to execute if the user tries to submit a Flash form with validation errors, as follows:

- If you specify one or more valid Flash expressions, Flash executes the expressions.
- If you omit the attribute, Flash displays a dialog box with all applicable error messages.
- If you specify **onError=""** (an empty string) Flash does not display any message, but does not submit the form.

Your ActionScript can use the errors variable to determine the fields and errors. The errors object has the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The <em>name</em> attribute of the control's CFML tag.</td>
</tr>
<tr>
<td>field</td>
<td>The internal name used by Flash for the field name (for example, _level0.field1).</td>
</tr>
<tr>
<td>value</td>
<td>The value in the field.</td>
</tr>
</tbody>
</table>
message  The message attribute of the control's CFML tag.

The following example shows cfform tags with an onError attribute that selects the tab in an accordion or tab navigator that contains a lastName field with an invalid entry:

```html
<cfform name="form1" format="flash" width="800" height="500"
onError="if (errors['lastName'] != undefined
  ){tabA.selectedIndex=0; _root.lastName.setFocus();}">
</cfform>
```

Incorporating HTML form tags and attributes

In HTML format, the cfform tag lets you incorporate the following standard HTML elements. They are not available in Flash format:

- Standard HTML form tag attributes and values. The attributes and values are included in the form tag that cfform outputs in the page. For example, you can use form tag attributes like target or onMouseOver with cfform.
- HTML tags that can ordinarily be put within the HTML form tag. For example, you can use the HTML input tag to create a submit button in a cfform, without the other features of cfinput:

```html
<cfform>
  <input type = "Submit" value = " update... ">
</cfform>
```

Using forms in cfdiv, cflayout, cffpod, and cfwindow controls

The cfdiv, cflayout, cffpod, and cfwindow tags create AJAX-based controls that can serve as containers for interactive forms. When you use such a structure, you do not want submitting form information to cause a new page to be displayed; instead, you want dynamic code to modify the existing page without causing a complete reload. You can do this by using the onSuccess and onError attributes. The function specified by the onSuccess attribute gets called if the form data is submitted successfully. This function is responsible for updating the pod, layout, or window to reflect the results of the submission, for example, to display additional data or pop up a confirmation window. This function must not take any arguments. The function specified by the onError attribute gets called if an error occurs when the form data is submitted. This function is responsible for handling the error, such as displaying an error message. This function must take two arguments: an error number and an error message.

Incorporating interactive fields in PDF forms

ColdFusion lets you use the cfform tag to create PDF forms that contain static and interactive form fields. The cfform tag must exist within a cfdocument tag (where format="pdf"). Only one cfform tag can exist within a cfdocument tag. Completed forms can be posted to the server as an HTTP Post, or the entire PDF can be submitted as binary stream. If the PDF is submitted, you can use the cffile tag to save completed PDF form to a hard drive: 

```html
<cffile action="write" file="c:\savedpdf.pdf" output="#PDF.content#">}
```

The output can be manipulated and extracted by using the tag. Only the following {{cfform attributes are supported in generating PDF forms:

- action
- format
To embed an existing PDF form generated by LiveCycle Designer or Acrobat, use the `cfpdf` tag.

**Example**
<h3>cfform Example</h3>

<!--- If Form.oncethrough exists, the form has been submitted. --->
<cfif IsDefined("Form.oncethrough")>

<cfif IsDefined("Form.testVal1")>

<h3>Results of Radio Button Test</h3>
Your radio button answer was yes
<cfelse>Your radio button answer was no
</cfif>
</cfif>

<h3>Results of Checkbox Test</h3>
<cfif IsDefined("Form.chkTest2")>
Your checkbox answer was yes
<cfelse>
Your checkbox answer was no
</cfif>
<cfif IsDefined("Form.textSample") AND Form.textSample is not ">
<h3>Results of Credit Card Input</h3>
Your credit card number, <cfoutput>#Form.textSample#</cfoutput>, 
was valid under the MOD 10 algorithm.
</cfif>
<cfif IsDefined("Form.sampleSlider")>
<h3>You gave this page a rating of #Form.sampleSlider#</h3>
</cfif>

<hr noshade="True">
</cfif>

<!--- Begin by calling the cfform tag. --->
<cfform name="cfformexample">

<h4>This example displays radio button input type for cfinput.</h4>
Yes <cfinput type = "Radio" name = "TestVal1" value = "Yes" checked>
No <cfinput type = "Radio" name = "TestVal1" value = "No">
<h4>This example displays checkbox input type for cfinput.</h4>
<cfinput type = "Checkbox" name = "ChkTest2" value = "Yes">
<h4>This shows client-side validation for cfinput text boxes.</h4>
(<i>This item is optional</i>)
<br>
Please enter a credit card number:
<cfinput type = "Text" name = "TextSample"
message = "Please enter a Credit Card Number"
validate = "creditcard" required = "No">
<h4>This example shows the use of the cfslider tag.</h4>
Rate your approval of this example from 1 to 10 by sliding control.
<br>
1 <cfslider name = "sampleSlider" width="100"
label = "Page Value: " range = "1,10"
message = "Please enter a value from 1 to 10" > 10
<p><cfinput type = "submit" name = "submit" value = "show me the result"/>
<cfinput type = "hidden" name = "oncethrough" value = "Yes"></p>
</cfform>

A simple XML form

The following example shows a simple XML-format form. It uses the default.xsl transform that is supplied with ColdFusion to generate the HTML output for display:
<cfform name="testXForm" format="XML" skin="basic">
  <!-- Use cfformgroup to put the first and last names on a single line. -->
  <cfformgroup type="horizontal">
    <cfinput type="text" name="firstname" label="First Name:" value="Robert">
    <cfinput type="text" name="lastname" label="Last Name:" value="Smith">
  </cfformgroup>
  <cfinput type="password" name="password" label="Password:" value="">
  <cfinput type="hidden" name="hidden" label="hidden:" value="">
  <cfselect name="state" style="width:200" label="State">
    <option>California</option>
    <option selected>Utah</option>
    <option>Iowa</option>
    <option selected>New York</option>
  </cfselect>
  <cftextarea name="description" label="Description:" rows="5" cols="40">
    this is sample text.</cftextarea>
</cfform>

A simple PDF form

<cfdocument format="pdf">
  <cfdocumentsection ../>
  ...
  <cfform type="html/xform">
    <cfinput type="textbox" name="employeeName" value="#fullName#" readonly="true">
    <cfinput type="textbox" name="employeeID" value="#id#" readonly>
    <cfselect name="contributionPercentage" options="#optionsStruct#" required="true">
      <cfinput type="submit" name="SubmitAsHTTPPost">
      <cfinput type="submit" name="SubmitAsPDF" submitType="PDF">
    </cfform>
    ...
    ...
  </cfdocumentsection ../>
</cfdocument>
### cffunction

#### Description

Defines a function that you can call in CFML. Required to define ColdFusion component methods.

#### History

ColdFusion 10: `restPath, httpMethod, produces, {{consumes}}`

ColdFusion 8:
- Added `returnFormat`, `secureJSON`, and `verifyClient` attributes
- Added `component` as a valid value for the `returnType` attribute.

ColdFusion MX 7: Added the `description` attribute, and added the XML value to the `returnType` attribute.

ColdFusion MX: Added this tag.

#### Category

**Extensibility tags**

#### Syntax

```cfml
<cffunction
    name = "method name"
    access = "method access"
    description = "function description"
    displayName = "name"
    hint = "hint text"
    output = "yes|no"
    returnFormat = "not specified|JSON|plain|WDDX"
    returnType = "data type"
    roles = "securityRoles"
    secureJSON = "yes|no"
    verifyClient = "no|yes">
```

#### See also

`cfargument, cfcomponent, cfinterface, cfinvoke, cfinvokeargument, cfobject, cfproperty, cfreturn, SerializeJSON`

#### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>A string; a component method that is used in the <code>cfcomponent</code> tag.</td>
</tr>
</tbody>
</table>
| **access** | Optional | **public** | The client security context from which the method can be invoked. The following values are valid:

- **private**: available only to the component that declares the method and any components that extend the component in which it is defined.
- **package**: available only to the component that declares the method, components that extend the component, or any other components in the package. Methods are available to the CFC pages in the same package.
- **public**: available to a locally executing page or component method.
- **remote**: available to a locally or remotely executing page or component method, or a remote client through a URL, Flash, or a web service. To publish the function as a web service, this option is required.

| **description** | Optional | Supplies a short text description of the function.

<p>| <strong>displayname</strong> | Optional | Meaningful only for CFC method parameters. A value to be displayed in parentheses following the function name when using introspection to show information about the CFC. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hint</td>
<td>Optional</td>
<td>Meaningful only for CFC method parameters. Text to be displayed when using introspection to show information about the CFC. The hint attribute value follows the syntax line in the function description.</td>
</tr>
<tr>
<td>output</td>
<td>Optional</td>
<td>Function body is processed as standard CFML. Specifies under which conditions the function can generate HTML output. The following values are valid:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- yes: the entire function body is processed as if it were in a cfoutput tag. Variables names surrounded by number signs (#) are automatically replaced with their values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- no: the function is processed as if it were within a cfsilent tag. If you do not specify this attribute, the function body is processed as standard CFML. Any variables must be in cf output tags.</td>
</tr>
<tr>
<td>returnformat</td>
<td></td>
<td>Return as WDDX or XML; see description. The format in which to return values to a remote caller. This attribute has no effect on values returned to a local caller. The following values are valid:</td>
</tr>
</tbody>
</table>
• `json`: serialize the return value into JSON format before returning it remotely.
• `wddx`: serialize the return value into WDDX format before returning it remotely.
• `plain`: ensure that the return value is a type that ColdFusion can convert directly to a string, and return the string value without serialization. Valid types include all simple types, such as numbers, and XML objects. If the return value is a complex type, such as an array, or a binary value, ColdFusion generates an error. If you specify a `returnType` attribute, its value must be `any`, `boolean`, `date`, `guid`, `numeric`, `string`, `uuid`, `variableName`, or `XML`; otherwise, ColdFusion generates an error.

By default, ColdFusion serializes all return types (including simple return types), except XML, into WDDX format, and returns XML data as XML text. You can also use `returnFormat` as an HTTP request parameter when calling a remote CFC function. This parameter has the same effect as the `returnType` attribute and overrides any `returnFormat` attribute value specified in the `cffunction` tag.
| returnType | Required for a web service; Optional, otherwise. | any | String; a type name; data type of the function return value:  
- any  
- array  
- binary  
- boolean  
- component: the return value must be a ColdFusion component.  
- date  
- guid: the argument must be a UUID or GUID of the form xxxxx xxx-xxxx-xxxx-xxxx-xxxx where each _x_ is a character that represents a hexadecimal number (0-9A-F).  
- numeric  
- query  
- string  
- struct  
- uuid: the argument must be a ColdFusion UUID of the form xxxxx xxx-xxxx-xxxx-xxxxxxxx xxxxxxxxxx where each _x_ is a character that represents a hexadecimal number (0-9A-F).  
- variableName: a string formatted according to ColdFusion variable naming conventions.  
- void: does not return a value.  
- xml: allows web service functions to return CFML XML objects and XML strings.
| roles   | Optional | "" (empty) | A comma-delimited list of ColdFusion security roles that can invoke the method. Only users who are logged in with the specified roles can execute the function. If this attribute is omitted, all users can invoke the method. |

- A component name: If the returnType attribute value is not one of the preceding items, ColdFusion treats it as the name of a ColdFusion component. When the function executes, it generates an error if the value that is returned is not a CFC with the specified name. **Note:** If a function does not return a value and the returnType value is `string`, ColdFusion generates an error; ColdFusion does not generate an error for other types.
| secureJSON | Optional | See Description | A Boolean value that specifies whether to add a security prefix in front of any value that the function returns in JSON-format in response to a remote call. The default value is the value of any This.secureJSON variable in the Application.cfc file or the secureJSON attribute of the cfapplication tag, or if there is no{{secureJSON}} application setting, the Prefix Serialized JSON setting in the Administrator Server Settings > Settings page, which defaults to false. For more information see Improving security in Ajax programming rules and techniques in the Developing ColdFusion Applications. |
| verifyClient | Optional | no | A Boolean value that specifies whether to require remote function calls to include an encrypted security token. For use with ColdFusion AJAX applications only. For more information see Improving security in Ajax programming rules and techniques in the Developing ColdFusion Applications. |
### restPath

**Optional**

Specify to use a sub-resource path for the CFC. The path is case-sensitive. Also, it is preferable to avoid special characters when you specify the path. You can include regular expressions in the path. For details of the expressions that you can specify, see the attribute `restPath` in `cfcomponent`.

### httpMethod

**The HTTP method to use, must be one of the following:**

- **GET**: Requests information from the server. Any data that the server requires to identify the requested information must be in the URL or in `{{cfhttp type="URL"}}` tag.
- **POST**: Sends information to the server for processing. Requires one or more `cfhttpparam` tags. Often used for submitting form-like data.
- **PUT**: Requests the server to store the message body at the specified URL. Use this method to send files to the server.
- **DELETE**: Requests the server to delete the specified URL.
**HEAD:** Identical to the `GET` method, but the server does not send a message body in the response. Use this method for testing hypertext links for validity and accessibility, determining the type or modification time of a document, or determining the type of server. If you have not specified `HEAD`, by default, `GET` method is called. However, message body is not sent in the response.

**OPTIONS:** A request for information about the communication options available for the server or the specified URL. This method enables the ColdFusion application to determine the options and requirements associated with a URL, or the capabilities of a server, without requesting any additional activity by the server. If you have not specified `OPTIONS`, then ColdFusion sends a response. Overrides the value that you specify at component level.
<table>
<thead>
<tr>
<th>produces</th>
<th>Optional</th>
<th>/</th>
</tr>
</thead>
</table>

Comma-separated list of acceptable MIME types. Used to specify the MIME media type of representation a resource can produce, for example, `{{produces="text/plain,text/html"}}`. If a resource can produce more than one MIME media type, the function chosen corresponds to the most acceptable media type as declared by the client. If no value is specified, `/` is taken by default, which means, all MIME types are consumed. Overrides the value that you specify at the component level.
consumes | Optional | / | Comma-separated list of acceptable MIME types, for example consumes="text/plain,text/html". Used to specify which MIME media types of representation a resource can accept, or consume, from the client. This attribute overrides the same attribute at the component level. If no value is specified, / is taken by default, which means, all MIME types are consumed. Adobe recommends that you avoid conflict scenarios while specifying the attributes produces and consumes. For example, avoid situations such as the following: In function 1, you specify produces as text/xml and consumes as text/ and in function 2, produces as text/ and consumes as text/xml. Here, both the functions are valid for the request with accept = text/xml and consumes as text/xml.

Usage

The cffunction tag can define a function that you call in the same manner as a ColdFusion built-in function. To define a ColdFusion component (CFC) method, use a cffunction tag. The following example shows cffunction tag attributes for a simple CFC method that returns a ColdFusion Query object.

```
<cffunction
    name="getEmployees"
    access="remote"
    returnType="query"
    hint="This query returns all records in the employee database. It can drill-down or narrow the search, based on optional input parameters.">
```

For detailed information on using the cffunction tag for ColdFusion components, see Building and Using ColdFusion Components in the Developing ColdFusion Applications.
Adobe ColdFusion Documentation

If you specify returnformat="json" and the function returns a query, ColdFusion serializes the query into a
JSON Object with two entries, and array of column names, and an array of column data arrays. For more
information see SerializeJSON.
If you specify a roles attribute, the function executes only if a user is logged in and belongs to one of the specified
roles.
If you specify variableName for the returnType attribute, the function must return a string that is in ColdFusion
variable name format; that is, the function must return a string that starts with a letter, underscore, or Unicode
currency symbol, and consist of letters, numbers, and underscores (_), periods, and Unicode currency symbols,
only. ColdFusion does not check whether the value corresponds to an existing ColdFusion variable.
Example

<cfcomponent>
<cffunction name="getEmp">
<cfquery
name="empQuery" datasource="ExampleApps" >
SELECT FIRSTNAME, LASTNAME, EMAIL
FROM tblEmployees
</cfquery>
<cfreturn empQuery>
</cffunction>
<cffunction name="getDept">
<cfquery name="deptQuery" datasource="ExampleApps" >
SELECT *
FROM tblDepartments
</cfquery>
<cfreturn deptQuery>
</cffunction>
</cfcomponent>

#back to top

© 2014 Adobe Systems Incorporated. All rights reserved.

548


Tags g-h

This section lists tags starting with alphabets G-H with their description.

- cfgraph
- cfgraphdata
- cfgrid
- cfgridcolumn
- cfgridrow
- cfgridupdate
- cfheader
- cfhtmlhead
- cfhtmltopdf
- cfhtmltopdfitem
- cfhttp
- cfhtpparam
**cfgraph**

**Description**

This tag is deprecated. Use the `cfchart`, `cfchartdata`, and `cfchartseries` tags instead. Displays data graphically.

**History**

**ColdFusion 11: Removed this tag.**

ColdFusion MX: Deprecated this tag. It works differently than it did in ColdFusion 5, and it might not work in later releases.

The incompatibilities between the ColdFusion MX implementation and earlier implementations of this tag are as follows:

<table>
<thead>
<tr>
<th>cfgraph tag attribute</th>
<th>ColdFusion MX functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Ignored.</td>
</tr>
<tr>
<td>Titlefont</td>
<td>Ignored.</td>
</tr>
<tr>
<td>Barspacing</td>
<td>Ignored.</td>
</tr>
<tr>
<td>Bordercolor</td>
<td>Color used for border, gridlines, and text displays.</td>
</tr>
<tr>
<td>Colorlist</td>
<td>List of colors to use for each data point for bar, pyramid, area, horizontalbar, cone, cylinder, step, and pie charts.</td>
</tr>
<tr>
<td>Valuelabelfont</td>
<td>Sets value label text font. If the Valuelabelfont, Itemlabelfont, and Legendfont values differ, ColdFusion uses the last value that you specify in the tag. Arial is not supported; it is mapped to Dialog.</td>
</tr>
<tr>
<td>Itemlabelfont</td>
<td>Sets item label text font. If the Valuelabelfont, Itemlabelfont, and Legendfont values differ, ColdFusion uses the last value that you specify in the tag. Arial is not supported; it is mapped to Dialog.</td>
</tr>
<tr>
<td>Legendfont</td>
<td>Sets legend text font. If the Valuelabelfont, Itemlabelfont, and Legendfont values differ, ColdFusion uses the last value that you specify in the tag. Arial is not supported; it is mapped to Dialog.</td>
</tr>
<tr>
<td>ShowLegend</td>
<td>• above, below, left, right: these options cause the legend to display, but have no effect on its location.</td>
</tr>
<tr>
<td></td>
<td>• none: prevents display of a legend.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Valuelabelsize</td>
<td>Sets value label text size. If the Valuelabelsize and Itemlabelsize values differ, ColdFusion uses the last value that you specify in the tag.</td>
</tr>
<tr>
<td>Itemlabelsize</td>
<td>Sets item label text size.</td>
</tr>
<tr>
<td>Itemlabelorientation</td>
<td>Ignored. ColdFusion calculates best orientation based on label and graph size.</td>
</tr>
</tbody>
</table>
| Borderwidth       | • a nonzero number: default-width border, regardless of number value.  
                   • 0: no border.                                                                                                               |
| Depth             | • 0: displays graph with two-dimensional appearance.  
                   • any other value: displays graph with three-dimensional appearance.                                                   |
| Linewidth         | Ignored.                                                                                                                                 |
| Showvaluelabel    | • yes: displays values on mouse-click.  
                   • no: suppresses value displays.  
                   • rollover: displays values on mouse-over.                                                                                 |
| Valuelocation     | Ignored.                                                                                                                                 |
| url               | URL of page to open if any item in the graph is clicked.  
                   The following variables may be used within the URL; they are substituted with real values before the URL is accessed:  
                   • "$value$": selected row/column value or an empty string.  
                   • "$itemlabel$": selected item (column) value or an empty string.  
                   • "$serieslabel$": selected series (row) value or an empty string.  
                   • "javascript:...": executes client side scripts. |
| Urlcolumn         | Ignored.                                                                                                                                 |
| Type="HorizontalBar" | The (0,0) coordinate is located at the lower left.                                                                                     |
| ScaleFrom         | If the smallest value in the data is less than scaleFrom or the largest value in the data is greater than scaleTo, the respective data value is used as the minimum or maximum on the Y scale. Therefore, regardless of the scaleFrom or scaleTo value, all data values display. |
**cfgraphdata**

**Description**

This tag is deprecated. Use the [cfchart](#), [cfchartdata](#), and [cfchartseries](#) tags instead. Displays a data point in a graph. Used within the [cfgraph](#) tag.

**History**

**ColdFusion 11: Removed this tag.**

ColdFusion MX: Deprecated this tag. It works differently than in ColdFusion 5 and might not work in later releases.
cfgrid

Description

Used in the cfform tag. Puts a grid control (a table of data) in a ColdFusion form. To specify grid columns and row data, use the cfgridcolumn and cfgridrow tags, or use the query attribute, with or without cfgridcolumn tags.

For CFC methods that returns numeric data with a leading zero, for example, zip code 02674, the zero is interpreted by the bind expression as an octal number and its decimal equivalent (in this case 1468) even if you set returnformat="string". To resolve this issue, for URL binds or binds routed by way of a JavaScript function (for example, using cfajaxproxy), you can set returnformat=plain to retain the numeric value. Also, leading zeros are stripped from the suggestion list for autosuggest controls.

Category

Forms tags

Syntax

```xml
<cfgrid
   name="name"
   align="value"
   appendKey="yes|no"
   autoWidth="yes|no"
   bgColor="web color"
   bind="bind expression"
   bindOnLoad="yes|no"
   bold="yes|no"
   colHeaderAlign="left|right|center"
   colHeaderBold="yes|no"
   colHeaderFont="font_name"
   colHeaderFontSize="size"
   colHeaderItalic="yes|no"
   colHeaders="yes|no"
   colHeaderTextColor="web color"
   collapsible="false|true"
   delete="yes|no"
   deleteButton="text"
   enabled="yes|no"
   font="column_font"
   fontSize="size"
   format="applet|Flash|html|xml"
   gridDataAlign="left|right|center"
   gridLines="yes|no"
   groupfield="column name"
   height="integer"
   highlightHref="yes|no"
   href="URL"
   hrefKey="column_name"
   hSpace="integer"
   insert="yes|no"
   insertButton="text"
   italic="yes|no"
   maxRows="number"
   multirowselect="yes|no"
   notSupported="text"
```
onBlur="ActionScript"
onChange="ActionScript or bind expression"
onError="JavaScript function name"
onFocus="ActionScript function"
onLoad="JavaScript function name"
onValidate="JavaScript function name"
pageSize="number of rows"
pictureBar="yes|no"
preservePageOnSort="yes|no"
query="query name"
rowHeaderAlign="left|right|center"
rowHeaderBold="yes|no"
rowHeaderFont="font name"
rowHeaderFontSize="size"
rowHeaderItalic="yes|no"
rowHeaders="yes|no"
rowHeaderTextColor="web color"
rowHeight="pixels"
selectColor="web color"
selectMode="mode"
selectOnLoad="yes|no"
sort="yes|no"
sortAscendingButton="text"
sortDescendingButton="text"
stripeRowColor="web color"
stripeRows="yes|no"
style= "style specification"
tintColor="web color"
title="text"
tooltip="text"
visible="yes|no"
vSpace="integer"
width="integer">

zero or more cfgridcolumn and cfgridrow tags
Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also
cfajaximport, cfapplet, cfcalendar, cfgridcolumn, cfgridrow, cfgridupdate, cfform, cfformgroup, cfformitem, cfinput, cfselect, cfslider, cftextarea, cftree, Using HTML grids in the Developing ColdFusion Applications

History

ColdFusion 9.0.1: Added the attribute multirowselect, supported only in HTML grids.

ColdFusion 9:

- Added collapsible, groupfield, onLoad, and title attributes, supported in HTML grids only.
- Added ability to use the insert attribute in HTML grids.

ColdFormation 8: Added support for HTML format grids, including the html value of the format attribute and the following attributes: bind, bindOnLoad, pageSize, preservePageOnSort, stripeRows, stripeRowColor. ColdFusion MX 7.01: Added support for the onBlur and onFocus events.

ColdFusion MX 7:

- Added the format attribute and support for Flash and XML output.
- Added enabled, onChange, style, tooltip, and visible attributes (Flash format only).

ColdFusion MX: Changed the rowHeaderWidth attribute: ColdFusion does not use the rowHeaderWidth attribute. You can omit it.

Attributes

Note

In XML format, ColdFusion passes all attributes to the XML. The supplied XSLT skins do not handle or display XML format grids, but do display applet and Flash format grids.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt; formats</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required; all</td>
<td></td>
<td>Name of the grid control.</td>
</tr>
<tr>
<td>align</td>
<td>Optional; applet</td>
<td>Alignment of the grid cell contents:</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>--------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Top</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Left</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Bottom</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Baseline</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Texttop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Absbottom</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Middle</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Absmiddle</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Right</td>
<td></td>
</tr>
</tbody>
</table>

| appendKey             | Optional; HTML, applet | yes | • yes: when used with href, appends "CFGRIDKEY=\" and information about the selected items. For details, see the section Using the href attribute. |
|                       |                 |     | • no |

<p>| autoWidth             | Optional; HTML, applet | no  | • yes: sets column widths so that all columns display within the grid width. Widths are equal or the proportions are determined by the relative cfgridcolumnwidth attribute values. Horizontal scroll bars are not available. |
|                       |                 |     | • no: sets columns to equal widths or the values specified in the cfgridcolumnwidth attributes. |</p>
<table>
<thead>
<tr>
<th><strong>bgColor</strong></th>
<th><strong>Optional; all</strong></th>
<th>Background color of the control. For most formats, can be a hexa-decimal format or a named color. For a hexadecimal value, use the form &quot;##xxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none. For a list of the supported named colors, see <a href="#">cfchart</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bind</strong></td>
<td><strong>Optional; HTML</strong></td>
<td>A bind expression used to fill the contents of the grid. Cannot be used with the <code>query</code> attribute. For more information, see <a href="#">Binding data to form fields</a> in <a href="#">Using Ajax Data and Development Features</a> in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Default</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| bindOnLoad       | Optional; HTML              | yes     | • **yes:** executes the bind attribute expression when first loading the form.  
• **no:** does not execute the bind attribute expression until the first bound event. Ignored if there is no bind attribute. 
For more information, see Using the bindOnLoad attribute in Using Ajax form controls and features in Developing ColdFusion Applications. |
| bold             | Optional; all               | no      | • **yes:** displays text in bold.  
• **no**                                                                                                                                                                                                  |
| colHeaderAlign   | Optional; applet            | left    | • **left:** left-aligns the column header text.  
• **right:** right-aligns the column header text.  
• **center:** centers the column header text.                                                                                                                                            |
| colHeaderBold    | Optional; all               | no      | • **yes:** displays column headers in bold.  
• **no**                                                                                                                                                                                                  |
| colHeaderFont    | Optional; all               |         | Font of column header.                                                                                                                                                                                      |
| colHeaderFontSize| Optional; all               |         | Size of column header text, in points.                                                                                                                                                                    |
| colHeaderItalic  | Optional; all               | no      | • **yes:** displays column headers in italic.  
• **no**                                                                                                                                                                                                  |
| colHeaders       | Optional; Applet, Flash     | yes     | • **yes:** displays column headers.  
• **no**                                                                                                                                                                                                  |
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| colHeaderTextColor | Optional; all                 |       | Color of column headers.  
  - Options: same as for `textColor` attribute. |
| collapsible        | Optional; HTML                | False | A Boolean value specifying whether the user can collapse the entire grid by clicking an arrow on the title bar. Specifying this attribute adds a title bar to the grid. |
| delete             | Optional; HTML, applet        | no    |  
  - yes: users can delete row data from the grid; takes effect only if `selectmode="edit"`.  
  - no                                                                                           |
| deleteButton       | Optional; HTML, applet        | Delete| Text for the Delete button; takes effect only if `selectmode="edit"`. |
| enabled            | Optional; Flash               | yes   | Flash format only: Boolean value that specifies whether the control is enabled. A disabled control appears in light gray. |
| font               | Optional; all                 |       | Font of text.                                                               |
| fontSize           | Optional; all                 |       | Size of text, in points.                                                   |
| format          | Optional; all | applet | • applet: generates a Java applet.  
• Flash: generates a Flash grid control.  
• html: generates an AJAX-based HTML grid control that supports data binding.  
• xml: generates an XML representation of the grid.  
In XML format forms, includes the generated XML in the form.  
In HTML format forms, puts the XML in a string variable with the name specified by the name attribute. |
|-----------------|---------------|--------|--------------------------------------------------------------------------------|
| gridDataAlign   | Optional; applet | left   | • left: left-aligns data within the column.  
• right: right-aligns data within the column.  
• center: centers data within the column. |
| gridLines       | Optional; applet, Flash | yes    | • yes: enables row and column rules.  
• no |

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupField</td>
<td>Optional; HTML</td>
<td>Don't group</td>
<td>Puts the grid rows into groups, organized by the column specified in this attribute. Each group is collapsible and has a header with the column name, group field value, and number of entries in the group. If you set this option, the column pull-down menu shows two grouping options: The show in Groups option turns column grouping on and off. The Group By This Field option sets the grouping to use the selected column. Users display the pull-down menu by moving the mouse over a column head and clicking the down arrow that appears. You can use this attribute with static grids only, do not use it with dynamic grids that get their data using bind expressions.</td>
</tr>
<tr>
<td>height</td>
<td>Optional; all</td>
<td>300 (applet only)</td>
<td>Height of the control, in pixels. If you omit the attribute in Flash format, the grid sizes automatically.</td>
</tr>
</tbody>
</table>
| highlightHref| Optional; applet  | yes             | • yes: highlights links associated with an `href` attribute value.  
• no |
<p>| href         | Optional; HTML,. applet |                | URL or name of a query column that contains URLs to hyperlink each grid cell with. |</p>
<table>
<thead>
<tr>
<th>hrefKey</th>
<th>Optional; HTML, applet</th>
<th>A query column to use for the value appended to the href URL of each cell, if appendKey=&quot;True&quot;. If you use cfgridcolumn tags, the column must be specified in one of these tags.</th>
</tr>
</thead>
<tbody>
<tr>
<td>hSpace</td>
<td>Optional; applet</td>
<td>Horizontal space to the left and right of the control, in pixels.</td>
</tr>
<tr>
<td>insert</td>
<td>Optional; applet, HTML</td>
<td>no • yes: users can insert row data in the grid; takes effect only if selectmode=&quot;edit&quot;. • no</td>
</tr>
<tr>
<td>insertButton</td>
<td>Optional; applet</td>
<td>Insert Text for the Insert button; takes effect only if selectmode=&quot;edit&quot;.</td>
</tr>
<tr>
<td>italic</td>
<td>Optional; all</td>
<td>no • yes: displays text in italic. • no</td>
</tr>
<tr>
<td>maxRows</td>
<td>Optional; all</td>
<td>Maximum number of rows to display in the grid.</td>
</tr>
<tr>
<td>multirowselect</td>
<td>Optional; HTML</td>
<td>no Allows selection of multiple rows. This is particularly useful in the cases where batch processing is required, for example, moving multiple records at a time. If yes, a check box appears as the first column of the grid, enabling selection of multiple records. A Select All/Deselect All option also appears.</td>
</tr>
</tbody>
</table>

Note
If multirowselect="yes" then a check box appears as the first column of the grid, enabling selection of multiple records. A Select All/Deselect All option also appears.
s", then row data is sent as an array of structs as opposed to a struct if multirowselect = "no". Also, if the grid data is manipulated by the user, for example, using JavaScript, to move records when a button is clicked, set the method to POST. This is required as a GET method imposes restrict
ions on the amount of data
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notSupported</td>
<td>Optional; applet</td>
<td>Text to display if the browser does not support Java or has Java support disabled. Default: &quot;&lt;b&gt;Browser must support Java to view ColdFusion Java Applets&lt;/b&gt;&quot;</td>
</tr>
<tr>
<td>onBlur</td>
<td>Optional, Flash</td>
<td>ActionScript that runs when the grid loses focus.</td>
</tr>
<tr>
<td>onChange</td>
<td>Optional; HTML, Flash</td>
<td>Flash format: ActionScript to run when the control changes due to user action in the control. HTML format: Required for HTML format grids that specify a <code>bind</code> attribute and a <code>selectMode</code> value of <code>edit</code>. A bind expression that calls a CFC method, JavaScript function, or URL to update the data source. If a URL is called, since the data is passed in JSON format to the URL page, use the function <code>DeserializeJS</code>. The arguments <code>cfgridchanget</code> and <code>idrow</code> must be serialized to JSON strings if a JavaScript bind is used to pass these arguments to a URL.</td>
</tr>
<tr>
<td>onError</td>
<td>Optional; HTML, applet</td>
<td>In HTML format grids, name of a JavaScript function to execute if an error occurs. In applet format grids, name of a JavaScript function to execute if validation fails.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>onFocus</td>
<td>Optional, Flash</td>
<td></td>
</tr>
<tr>
<td>onLoad</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>onValidate</td>
<td>Optional; applet</td>
<td></td>
</tr>
<tr>
<td>pageSize</td>
<td>Optional; HTML</td>
<td>10</td>
</tr>
</tbody>
</table>
| pictureBar   | Optional; applet | no | • yes: puts images (and no text) on the Insert, Delete, and Sort buttons.  
• no: puts text (and no images) on the Insert, Delete, and Sort buttons. |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>preservePageOnSort</td>
<td>Optional; HTML</td>
<td>no</td>
<td>Specifies whether to display the page with the current page number, or display page 1, after sorting (or resorting) the grid. If this attribute is yes, selections are preserved when the grid sorts.</td>
</tr>
<tr>
<td>query</td>
<td>Optional; all</td>
<td></td>
<td>Name of the query associated with the control. Cannot be used with the bind attribute.</td>
</tr>
<tr>
<td>rowHeaderAlign</td>
<td>Optional; applet</td>
<td>left</td>
<td>• left: left-aligns the row header text.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• right: right-aligns the row header text.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• center: centers the row header text.</td>
</tr>
<tr>
<td>rowHeaderBold</td>
<td>Optional; applet</td>
<td>no</td>
<td>• yes: displays row label text in bold.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no</td>
</tr>
<tr>
<td>rowHeaderFont</td>
<td>Optional; applet</td>
<td></td>
<td>Font for the row labels.</td>
</tr>
<tr>
<td>rowHeaderFontSize</td>
<td>Optional; applet</td>
<td></td>
<td>Text size of the row labels, in points.</td>
</tr>
<tr>
<td>rowHeaderItalic</td>
<td>Optional; applet</td>
<td>no</td>
<td>• yes: displays row label text in italic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no</td>
</tr>
<tr>
<td>rowHeaders</td>
<td>Optional; applet</td>
<td>yes</td>
<td>• yes: displays a column of numeric row labels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no</td>
</tr>
<tr>
<td>rowHeaderTextColor</td>
<td>Optional; applet</td>
<td>black</td>
<td>Text color of grid control row headers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Options: same as for the textColor attribute.</td>
</tr>
<tr>
<td>rowHeight</td>
<td>Optional; Applet, Flash, XML</td>
<td></td>
<td>Minimum row height, in pixels. Used with cfgrid column type = &quot;Image&quot; ; defines space for graphics to display in row.</td>
</tr>
<tr>
<td><strong>selectColor</strong></td>
<td>Optional; all</td>
<td></td>
<td>Background color for a selected item.</td>
</tr>
<tr>
<td><strong>selectMode</strong></td>
<td>Optional; all</td>
<td></td>
<td>Applet format: Browse; HTML, Flash format: Row</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>Edit</strong>: the user can edit grid data. Selecting a cell lets the user edit the cell.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>Row</strong>: user selections automatically extend to the row that contains selected cell. The following are used in applet format only; HTML and Flash formats interpret these as Row:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>Single</strong>: user selections are limited to the selected cell.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>Column</strong>: user selections automatically extend to the column that contains selected cell.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>Browse</strong>: the user can only browse grid data.</td>
</tr>
<tr>
<td><strong>selectOnLoad</strong></td>
<td>Optional; HTML</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
</table>
| sort                   | Optional; applet          | no               | Adds sort buttons to perform simple text sorts on a user-selected column:  
- **yes**: put sort buttons on the grid control.  
- **no**: Independent of this setting, users can sort columns by clicking the column head. If select Mode="browse", the table cannot be sorted. |
| sortAscendingButton    | Optional; applet          | A > Z            | Text for the Sort button. |
| sortDescendingButton   | Optional; applet          | Z > A            | Text for the Sort button. |
| stripeRowColor         | Optional; HTML            |                  | The color to use for one of the alternating stripes. The bgColor setting determines the other color. |
| stripeRows             | Optional; HTML            | no               | Boolean value that indicates whether to make the rows stripes in alternating colors. |
| style                  | Optional; Flash           |                  | Must be a style specification in CSS format. Ignored for type= "text". |
| target                 | Optional; HTML, applet    |                  | The target frame or window in which to display the href URL; for example, "_blank". |
| **textContent** | Optional; Flash, applet | Color of text. Can be a hexadecimal value or a named color. For a hexadecimal value, use the form "##xxxxxx", where x = 0-9 or A-F; use two number signs or none. For a list of the supported named colors, see [cfchart](#). |
| **title** | Optional; HTML | Text to display as a title at the top of the grid. Specifying this attribute adds a title bar to the grid. |
| **tooltip** | Optional; Flash | Flash format only: text to display when the mouse pointer hovers over the control. |
| **visible** | Optional; Flash | yes |
| **vSpace** | Optional; applet | Vertical space above and below the control, in pixels. |
| **width** | Optional; all | 300 (applet only) |

**Usage**

Most of the following paragraphs describe grid features that apply to all, or at least two, grid formats. For information that is specific to Flash forms, see [Creating Forms in Flash](#) in the [Developing ColdFusion Applications](#). For information that is specific to HTML format grids, see [Using HTML grids](#) in the [Developing ColdFusion Applications](#). This tag must be in a `<cfform>` tag block. An applet format grid requires the client to download a Java applet. Also, if the client does not have an up-to-date Java plug-in installed, the system might also have to download an updated...
Java plug-in to display an applet format grid. A Flash format grid generates a Flash control, and can be embedded in an HTML format `<cfform>` tag. For this tag to work properly in either Flash or applet format, the browser must also be JavaScript-enabled.

**Note**

If you specify Flash format for this tag in an HTML format form, and you do not specify `height` and `width` attributes, Flash takes up more than the remaining visible area on the screen. If any other output follows the grid, including any form controls, users must scroll to see it. Therefore, if you follow a Flash grid in an HTML form with additional output, specify `height` and `width` values.

You can populate a `<cfgrid>` with data from a `<cfquery>`. If you do not specify any `<cfgridcolumn>` tags in the `<cfgrid>` body, ColdFusion generates a grid with the following:

- A column for each column in the query.
- A default header for each column, created by replacing hyphen or underscore characters in the table column name with spaces. The first character, and any character after a space, are changed to uppercase; all other characters are lowercase.

This tag requires an end tag.

**Note**

Clicking the submit button while editing a grid cell occasionally causes the cell changes to be lost. To ensure that changes are submitted properly, Adobe recommends that after user updates data in a cell, they click another cell before submitting the form.

Returning `<cfgrid>` data to the action page

The following information applies to all `<cfgrid>` formats. Also, HTML format grids can dynamically get data by using a bind expression. For more information, see *Using HTML grids* in the *Developing ColdFusion Applications*. When a user submits a form, the `<cfgrid>` tag sends information about user actions by setting form variables in the data submitted to the form's action page. Because the data can vary, depending on the tag's `SelectMode` attribute value, the form variables that are returned also vary depending on this value. In general, the data returned falls into one of these categories:

- Simple data, returned from simple select operations
- Complex data, returned from insert, update, and delete operations

**Simple selection data (SelectMode = Single, Column, or Row)**

The data that form variables return to the `<cfform>`'s action page contains information about which cells the user selected. In general, ColdFusion makes this data available in the action page, as ColdFusion variables in the Form scope, with the naming convention `form.#GridName#.Column#`. Each `{SelectMode returns these form variables:}`
Complex update data (SelectMode = Edit)

The grid returns a large amount of data, to inform the action page of inserts, updates, or deletes that the user made to the grid. In most cases, you can use the `cfgridupdate` tag to automatically gather the data from the form variables; the tag collects data, writes SQL calls, and updates the data source. If you cannot use `cfgridupdate` (if, for example, you must distribute the returned data to more than one data source), write code to read form variables. In this mode, ColdFusion creates the following array variables in the Form scope for each `cfgrid`:

```coldfusion
form.#GridName#.ColumnName
form.#GridName#.original.ColumnName
form.#GridName#.RowStatus.Action
```

Each table row that contains an update, insert, or deletion has a parallel entry in each of these arrays. To view all the information for all the changes, you can traverse the arrays, as in this example. To make it work with a `cfgrid` on a submitted `cfform`, set the GridName variable to the name of the grid and the ColNameList to a list of the grid columns.
<cfloop index="ColName" list="#ColNameList#">
  <cfif IsDefined("form.#GridName#.ColName#")>
    <cfoutput><br>form.#GridName#.ColName#:<br></cfoutput>

    <cfset Array_New = form[#GridName#][ColName#]>
    <cfset Array_Orig = form[#GridName#][original][ColName#]>
    <cfset Array_Action = form[#GridName#]RowStatus.Action>

    <cfif NOT IsArray(Array_New)>
      <b>The form variable is not an array!</b><br>
    <cfelse>
      <cfset size = ArrayLen(Array_New)>
      <cfoutput>
        Result Array Size is #size#. <br>
        Contents:<br>
      </cfoutput>

      <cfif size IS 0>
        <b>The array is empty.</b><br>
      <cfelse>
        <table BORDER="yes">
          <tr>
            <th>Loop Index</th>
            <th>Action</th>
            <th>Old Value</th>
            <th>New Value</th>
          </tr>
          <cfloop index="LoopCount" from="1" to=#size#>
            <cfset Val_Orig = Array_Orig[LoopCount#]>
            <cfset Val_New = Array_New[LoopCount#]>
            <cfset Val_Action = Array_Action[LoopCount#]>
            <cfoutput>
              <tr>
                <td>#LoopCount#</td>
                <td>#Val_Action#</td>
                <td>#Val_Orig#</td>
                <td>#Val_New#</td>
              </tr>
            </cfoutput>
          </cfloop>
        </table>
      </cfif>
    </cfif>
  </cfif>
</cfloop>

Using the href attribute

When specifying a URL with grid items using the href attribute, the selectMode attribute value determines whether the appended key value is limited to one grid item or extends to a grid column or row. When a user clicks a linked grid item, a cfgridkey variable is appended to the URL, in this form:
If the `appendKey` attribute is set to no, no grid values are appended to the URL. The value of `selection` is determined by the value of the `selectMode` and attribute:

- If you specify a `hrefKey` attribute, `selection` is the field value of the column specified by the attribute. Otherwise, it is one of the following:
  - If `selectMode="Single"`, `selection` is the value of the column clicked.
  - If `selectMode="Row"`, `selection` is a comma-delimited list of column values in the clicked row, beginning with the value of the first cell in the row.
  - If `selectMode="Column"`, `selection` is a comma-delimited list of row values in the clicked column, beginning with the value of the first cell in the column.

When you use an `href` attribute, you can also specify a `target` attribute with any of the standard HTML target specifiers, `_blank`, `_parent`, `_self`, and `_top`, or with a specific frame name.

Enhancements made in ColdFusion 9.0.1

- In ColdFusion 9, data for the first row is available on form submission in a form with dynamic grid. In ColdFusion 9.0.1, the data is not available.
- If the type is `Boolean` and `selectmode` is `browse`, or `select=false`, the column is shown as a check box where click does not take effect.

Example

The following example creates a Flash form that displays a set of available courses from the CourseList table in the `cfdocexamples` database. For more complex examples that use the `cfgrid` tag, see `cfgridcolumn`, `cfgridrow`, and `cfgridupdate`.

```coldfusion
<!--- Query the database to fill up the grid. --->
<cfquery name = "GetCourses" dataSource = "cfdocexamples">
  SELECT Course_ID, Dept_ID, CorNumber, CorName, CorLevel
  FROM CourseList
  ORDER by Dept_ID ASC, CorNumber ASC
</cfquery>

<h3>cfgrid Example</h3>
<i>Currently available courses</i>
<!--- cfgrid must be inside a cfform tag. --->
<cfform>
  <cfgrid name = "FirstGrid" format="Flash"
    height="320" width="580"
    font="Tahoma" fontsize="12"
    query = "GetCourses">
  </cfgrid>
</cfform>
```
cfgridcolumn

Description

Used with the cfgrid tag in a cfform. Formats a column and optionally populates the column from a query. The font and alignment attributes used in cfgridcolumn override global font or alignment settings defined in cfgrid.

Category

Forms tags

Syntax

```
<cfgridcolumn
  autoExpand = "yes|no"
  name = "column name"
  bgColor = "web color|expression"
  bold = "yes|no"
  dataAlign = "left|right|center"
  display = "yes|no"
  font = "column font"
  fontSize = "size"
  header = "header"
  headerAlign = "left|right|center"
  headerBold = "yes|no"
  headerFont = "font name"
  headerFontSize = "size"
  headerIcon = "icon path"
  headerItalic = "yes|no"
  headerMenu = "yes|no"
  headerTextColor = "web color"
  href = "URL"
  hrefKey = "column name"
  italic = "yes|no"
  mask = "format mask"
  numberFormat = "format"
  select = "yes|no"
  target = "URL target"
  textColor = "web color|expression"
  type = "type"
  values = "comma-separated strings and/or numeric range"
  valuesDelimiter = "delimiter character"
  valuesDisplay = "comma-separated strings and/or numeric range"
  width = "column width">
```

⚠️ Note

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also
**History**

ColdFusion 9.0.1: Added the attributes `autoExpand` and `headerMenu` supported only in HTML grids.

ColdFusion 9: Added `boolean`, `date`, `numeric`, and `string_noCase` to the `type` attribute values supported in HTML grids.

ColdFusion MX 7: Added the `mask` attribute, and the currency `type` attribute value. ColdFusion MX: Changed behavior if `select="no"`: a user cannot select and edit the cell data, regardless of the `cfgrid selectmode` attribute value. When clicked, the cell border (and, depending on the `selectColor` value, the cell background) changes color, but the cell data cannot be edited.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt; formats</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoExpand</td>
<td>Optional; HTML</td>
<td>yes for first column and no for remaining columns</td>
<td>On a particular column, it lets you expand the specified column. Setting <code>autoExpand=&quot;yes&quot;</code> for multiple columns results in error. Also, if the attribute display is set to no, then <code>autoExpand</code> cannot be yes; else, it results in error.</td>
</tr>
<tr>
<td>name</td>
<td>Required; all</td>
<td></td>
<td>Name of the grid column element. If the grid uses a query, this attribute must be the name of the query column that populates the grid column.</td>
</tr>
<tr>
<td>bgColor</td>
<td>Optional; all</td>
<td></td>
<td>Color of background of grid column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Options: same as for the <code>textColor</code> attribute.</td>
</tr>
<tr>
<td>bold</td>
<td>Optional; all</td>
<td>As specified by <code>cfgrid</code></td>
<td>• yes: displays grid control text in bold.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no</td>
</tr>
</tbody>
</table>
| dataAlign | applet, Flash, HTML | As specified by cfgrid | Column data alignment:
- left
- right
- center |
|-----------|---------------------|------------------------|--------------------------|
| display   | all                 | yes                    | yes: displays header in bold.
no: hides the column. |
| font      | all                 | As specified by cfgrid | Font of data in column. |
| fontSize  | all                 | As specified by cfgrid | Size of text in column. |
| header    | all                 | yes                    | Text for the column header. Used only if the cfgrid colHeaders attribute is yes. The default value is yes. |
| headerAlign | applet | As specified by cfgrid | Column header text alignment:
- left
- right
- center |
| headerBold | HTML, applet | As specified by cfgrid | yes: displays header in bold.
no |
<p>| headerFont | HTML, applet | As specified by cfgrid | Font for the column header. |
| headerFontSize | HTML, applet | As specified by cfgrid | Size of text for the column header, in pixels. |
| headerIcon | Optional | | Location of an image file to use as the icon for header column of the grid. |
| {{headerMenu}} | Added in ColdFusion 9.0.1 | HTML | no |
|           |                     |                         | Lets you turn on/off the header menu of the grid column. Header menu is the drop-down list that appears on grid header columns on mouse hover. The attribute is helpful when you have images for grid headers. |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headerItalic</td>
<td>Optional; HTML, applet</td>
<td>As specified by cfgrid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: displays column header in italic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no.</td>
</tr>
<tr>
<td>headerTextColor</td>
<td>Optional; HTML, applet</td>
<td>Color of grid control column header text.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Options: same as for the textColor attribute.</td>
</tr>
<tr>
<td>href</td>
<td>Optional; HTML, applet</td>
<td>URL or query column name that contains a URL to hyperlink each grid column with.</td>
</tr>
<tr>
<td>hrefKey</td>
<td>Optional; HTML, applet</td>
<td>The query column to use for the value appended to the href URL of each column, instead of the column's value.</td>
</tr>
<tr>
<td>italic</td>
<td>Optional; all</td>
<td>As specified by cfgrid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: displays grid control text in italic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no.</td>
</tr>
</tbody>
</table>
| mask | Optional; Flash, HTML | A mask pattern that controls the character pattern that the form displays or allows users to input and sends to ColdFusion. For columns with the currency type attribute, the mask specifies the currency symbol. ColdFusion automatically inserts the character before the numeric value. For columns with text or numeric values, mask specifies the format to display or allow users to input, as follows:

- \( A = A-Za-z \)
- \( X = A-Za-z0-9 \)
- \( 9 = 0-9 \)
- \( ? = \) Any character
- All other characters = ColdFusion inserts the literal character.

If the column values are dates or timestamps, ColdFusion uses the mask pattern to format the selected date. For details of the date/time mask format, see the section date/time formats in mask attribute. Masking supports HTML grids. The default format is \( m/d/y \), for example 05/06/75, where \( m \) is month with leading zeros, \( d \) is day with leading zeros, and \( y \) is two-digit representation of year. For further details, go to the following URL: [http://www.extjs.com/deploy/dev/docs/output/Date.html](http://www.extjs.com/deploy/dev/docs/output/Date.html) |
<table>
<thead>
<tr>
<th>attribute</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numberFormat</td>
<td>Optional;Applet</td>
<td>Format for displaying numeric data in the grid. See the following table of numberFormat attribute mask characters.</td>
</tr>
<tr>
<td>select</td>
<td>Optional;all</td>
<td>yes</td>
</tr>
<tr>
<td>select</td>
<td></td>
<td>Determines selection behavior if the cfgrid selectmode attribute value is column, edit, or single; ignored for row or browse values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: users can select the column or select or edit cells in the column, as specified by the selectmode attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no: users cannot select the column or select or edit cells in the column.</td>
</tr>
<tr>
<td>target</td>
<td>Optional;HTML, Applet</td>
<td>Frame or standard HTML target in which to open link specified in href.</td>
</tr>
</tbody>
</table>
| textColor | Optional: Applet, Flash, HTML | Color of grid element text in column as a hexadecimal number or text name. To enter a hexadecimal value, use the form "##xx xxxx", where x = 0-9 or A-F; use two number signs or none. Limitations: In HTML format, must specify a valid HTML color. In Applet format, must be one of the following:  
- Any color, in hexadecimal format  
- Black  
- Red  
- Blue  
- Magenta  
- Cyan  
- Orange  
- Darkgray  
- Pink  
- Gray  
- White  
- Lightgray  
- Yellow |
|---|---|---|
| type | Optional: all | You can specify the following values in all formats:  
- **boolean**: column displays as check box; if cell is editable, user can change the check mark. In an onchange event, for static and dynamic grids, the data that is passed is converted to the format in which the boolean values are represented in the database.  
- **combobox**: displays a drop-down list with the values you specify for the attributes values and valuedisplay as options. |
- **numeric**: user can sort grid data numerically. In HTML format, if the cell is editable, the user can enter numeric values.
- **string_noCase**: user can sort grid data as case-insensitive text. In HTML format, if the cell is editable, the user can enter text values.

You can specify the following value in applet and Flash formats; it does not work in HTML format:

- **image**: grid displays the image specified by the URL in the column. If you use a relative URL, the image must be in the CFIDE/classes directory or a subdirectory. If the image is larger than the column cell, it is clipped to fit. Flash images must be JPEG files. Applet images can be JPEG or GIF files.

You can specify the following value in Flash format; it does not work in applet or HTML format:

- **image**: you can use the following built-in ColdFusion image names, in addition to paths to image files, in the column values: cd, computer, document, element, folder, floppy, fixed, remote.

You can specify the following value in Flash format; it does not work in applet or HTML format:

- **image**: you can use the following built-in ColdFusion image names, in addition to paths to image files, in the column values: cd, computer, document, element, folder, floppy, fixed, remote.
- **currency:** formats the column data as currency, aligning it around the decimal point. If users sort the grid by using this column, it sorts correctly for the currency. Use the `mask` attribute to specify a currency symbol; the default value is the dollar sign ($). You can specify the following value in HTML format; it does not work in applet or Flash format:

  ```html
  {{mask = "\$"}}
  ```

- **date:** The column contains date values. If the grid `selectMode` attribute value is `edit`, the cell is editable. When you click an editable cell, an icon appears that you can click to open a date picker and select a date.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Optional; HTML, applet</th>
<th>Formats cells in column as drop-down list boxes; specify items in drop-down list, for example:</th>
</tr>
</thead>
</table>
| `values`       |                        | ```
|                |                        | {{values = "arthur, scott, charles, 1-20, mabel"}} ``` |
| `valuesDelimiter` | Optional; HTML, applet | , (comma)                                                                                       |
| `valuesDisplay` |                        | Delimiter in `values` and `valuesDisplay` attributes                                              |
The following matrix describes the behavior of `type="boolean"`.

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>true (for static grids)</td>
<td>false (for static grids)</td>
</tr>
<tr>
<td>true (for dynamic grids)</td>
<td>NO (for dynamic grids)</td>
</tr>
<tr>
<td>For non-boolean or null</td>
<td>Y</td>
</tr>
</tbody>
</table>

In applet format only, you can use the following `numberFormat` attribute mask characters to format output in U.S. numeric and currency styles. For more information on using these mask characters, see `NumberFormat`. (The `cfgridcolumn` tag does not support international number formatting.)

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>(Underscore) Digit placeholder.</td>
</tr>
<tr>
<td>9</td>
<td>Digit placeholder.</td>
</tr>
<tr>
<td>.</td>
<td>(Period) Location of mandatory decimal point.</td>
</tr>
<tr>
<td>0</td>
<td>Located to left or right of mandatory decimal point; pads with zeros.</td>
</tr>
<tr>
<td>( )</td>
<td>Puts parentheses around mask if number is less than 0.</td>
</tr>
<tr>
<td>+</td>
<td>Puts plus sign before positive numbers, minus sign before negative numbers.</td>
</tr>
<tr>
<td>-</td>
<td>Puts space before positive numbers, minus sign before negative numbers.</td>
</tr>
</tbody>
</table>
(Comma) Separates every third decimal-place with a comma.

L,C
Left-justify or center-justify number within width of mask column. First character of mask must be L or C. Default: right-justified.

$ Puts dollar sign before formatted number. Must be the first character of mask.

^ (Caret) Separates left from right formatting.

date/time formats in mask attribute

By default, Flash displays date/time values in grid columns with a format that shows values such as Oct 29 2004 11:03:21. Use the mask attribute to display the date or time in a different format, as described in the following table:

<table>
<thead>
<tr>
<th>Pattern letter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Year. If the number of pattern letters is two, the year is truncated to two digits; otherwise, it appears as four digits. The year can be zero-padded, as the third example shows in the following set of examples: Examples:YY = 03YYYY = 2003YYYYY = 02003</td>
</tr>
</tbody>
</table>
| M              | Month in year. The format depends on the following criteria:  
  * If the number of pattern letters is one, the format is interpreted as numeric in one or two digits.  
  * If the number of pattern letters is two, the format is interpreted as numeric in two digits.  
  * If the number of pattern letters is three, the format is interpreted as short text.  
  * If the number of pattern letters is four, the format is interpreted as full text.  
  Examples:M = 7MM= 07MMM=JulMMMM= July |
<p>| D              | Day in month. Examples:D=4DD=04DD=10 |</p>
<table>
<thead>
<tr>
<th>Character</th>
</tr>
</thead>
</table>
| E         | Day in week. The format depends on the following criteria:  
|           | • If the number of pattern letters is one, the format is interpreted as numeric in one or two digits.  
|           | • If the number of pattern letters is two, the format is interpreted as numeric in two digits.  
|           | • If the number of pattern letters is three, the format is interpreted as short text.  
|           | • If the number of pattern letters is four, the format is interpreted as full text.  
|           | Examples: E = 1EE = 01EEE = MonEEE = Monday  
| A         | AM/PM indicator.  
| J         | Hour in day (0-23).  
| H         | Hour in day (1-24).  
| K         | Hour in am/pm (0-11).  
| L         | Hour in am/pm (1-12).  
| N         | Minute in hour. Examples: N = 3NN = 03  
| S         | Second in minute.  

Other text

You can add other text into the pattern string to further format the string. You can use punctuation, numbers, and all lowercase letters. Avoid upper case letters because they may be interpreted as pattern letters. Example: EEEE, MMM, D, YYYY at H:NN A = Tuesday, Sept. 8, 2003 at 1:26 PM

Example

The following example lets you update certain fields of the CourseList table in the cfdocexamples database. It uses cfgridcolumn tags to structure the table.
<!--- If the gridEntered field exists, the form has been submitted. --->
<cfif IsDefined("form.gridEntered")>
<cfgridupdate grid = "FirstGrid" dataSource = "cfdocexamples"
    tableName = "CourseList" keyOnly = "Yes">
</cfif>

<!--- Query the database to fill up the grid. --->
<cfquery name = "GetCourses" dataSource = "cfdocexamples">
    SELECT Course_ID, Dept_ID, CorNumber, CorName, CorLevel, CorDesc
    FROM CourseList
    ORDER by Dept_ID ASC, CorNumber ASC
</cfquery>

<html>
<head>
<title>cfgrid Example</title>
</head>
<body>
<h3>cfgrid Example</h3>
<i>You can update the Name, Level, and Description information for courses.</i>
<!--- The cfform tag must surround a cfgrid control. --->
<cfform action = "#CGI.SCRIPT_NAME#">
<cfgrid name = "FirstGrid" width = "500"
    query = "GetCourses" colheaderbold="Yes"
    font = "Tahoma" rowHeaders = "No"
    selectColor = "Red" selectMode = "Edit" >
    <!--- cffgridcolumn tags arrange the table and control the display. --->
    <!--- Hide the primary key, required for update --->
    <cffgridcolumn name = "Course_ID" display = "No">
    <!--- select="No" does not seem to have any effect!!! --->
    <cffgridcolumn name = "Dept_ID" header = "Department" Select="No" width="75"
        textcolor="blue" bold="Yes">
    <cffgridcolumn name = "CorNumber" header = "Course ##" Select="No" width="65">
    <cffgridcolumn name = "CorName" header = "Name" width="125">
    <cffgridcolumn name = "CorLevel" header = "Level" width="85">
    <cffgridcolumn name = "CorDesc" header = "Description" width="125">
    </cffgrid>
    <br>
    <cfinput type="submit" name="gridEntered">
</cfform>
</body>
</html>
**cfgridrow**

**Description**

Lets you define a `cfgrid` control that does not use a query as source for row data. If a query attribute is specified in the `cfgrid` tag, the `cfgridrow` tags are ignored.

**Category**

**Forms tags**

**Syntax**

```xml
<cfgridrow
   data = "col1, col2, ...
   delimiter = "delimiter character">
```

**Note**

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

**See also**

`cfgrid`, `cfgridcolumn`, `cfgridupdate`, `cfform`, `cfinput`, `cfselect`, `cfslider`, `cftextarea`, `cftree`

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Required</td>
<td></td>
<td>Delimited list of column values. If a value contains a delimiter character, it must be escaped with another delimiter character.</td>
</tr>
<tr>
<td>delimiter</td>
<td>Optional</td>
<td>, (comma)</td>
<td>Separator between column values.</td>
</tr>
</tbody>
</table>

**Example**

The following example shows how you use the `cfgridrow` tag can populate a `cfgrid` tag from list data:
<!--- Set two lists, each with the data for a grid column. --->
<cfset cities = "Rome,Athens,Canberra,Brasilia,Paris">
<cfset countries = "Italy,Greece,Australia,Brazil,France">
<cfform name = "cities">
<cfgrid name="GeoGrid" autowidth = "yes" vspace = "4" height = "120"
   font="tahoma" rowheaders="no">
  <cfgridcolumn name="City" header="City">
  <cfgridcolumn name="Country" header="Country">
  <!--- Loop through the lists using cfgridrow to populate the grid. --->
  <cfloop index="i" from="1" to="#ListLen(cities)#">
    <cfgridrow data="#ListGetAt(cities, i)#,#ListGetAt(countries, i)#"></cfgridrow>
  </cfloop>
</cfform>
**cfgridupdate**

**Description**

Used with a [cfgrid](#) tag. Updates data sources directly from edited grid data. This tag provides a direct interface with your data source. This tag applies delete row actions first, then insert row actions, then update row actions. If it encounters an error, it stops processing rows.

**Category**

Forms tags

**Syntax**

```xml
<cfgridupdate
    grid = "grid name"
    dataSource = "data source name"
    tableName = "table name"
    keyOnly = "yes|no">
    password = "data source password"
    tableOwner = "table owner"
    tableQualifier = "qualifier"
    username = "data source user name">
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

[cfgrid](#), [cfgridcolumn](#), [cfgridrow](#), [cfform](#), [cfapplet](#), [cfinput](#), [cfselect](#), [cfslider](#), [cftextinput](#), [cf](#), [tree](#)

**History**

ColdFusion 10: Added the clientInfo attribute.ColdFusion MX: Deprecated the connectString, dbName, dbServer, dbtype, provider, and providerDSN attributes. They do not work, and might cause an error, in releases later than ColdFusion 5.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientInfo</td>
<td>Optional</td>
<td></td>
<td>Structure containing properties of the client to be set on the database connection.</td>
</tr>
<tr>
<td>grid</td>
<td>Required</td>
<td>Name of the cfgrid form element that is the source for the update action.</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>-------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>dataSource</td>
<td>Required</td>
<td>Name of the data source for the update action.</td>
<td></td>
</tr>
<tr>
<td>tableName</td>
<td>Required</td>
<td>Name of the table to update. For ORACLE drivers, entry must be upper-case. For Sybase driver, entry is case sensitive; must be same case as used when table was created.</td>
<td></td>
</tr>
<tr>
<td>keyOnly</td>
<td>no</td>
<td>Applies to the update action:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: the WHERE criteria are limited to the key values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no: the WHERE criteria include key values and the original values of changed fields.</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td>Overrides password value specified in ODBC setup.</td>
<td></td>
</tr>
<tr>
<td>tableOwner</td>
<td>Optional</td>
<td>Table owner, if supported.</td>
<td></td>
</tr>
<tr>
<td>tableQualifier</td>
<td>Optional</td>
<td>Table qualifier, if supported. Purpose:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SQL Server and Oracle driver: name of database that contains the table.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Intersolv dBASE driver: directory of DBF files.</td>
<td></td>
</tr>
<tr>
<td>username</td>
<td>Optional</td>
<td>Overrides username value specified in ODBC setup.</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

The following example lets you update a database by using a cfgrid tag to add and delete entire records or to update the data in individual cells. The cfgridupdate tag processes the data from the submitted form and
updates the database.

<!---- If the gridEntered form field exists, the form was submitted. Perform gridupdate. --->
<cfif IsDefined("form.gridEntered") is True>
  <cfgridupdate grid = "FirstGrid" dataSource = "cfdocexamples" Keyonly="true"
    tableName = "CourseList">
  </cfif>

<!--- Query the database to fill up the grid. --->
<cfquery name = "GetCourses" dataSource = "cfdocexamples">
  SELECT Course_ID, Dept_ID, CorNumber, CorName, CorLevel, CorDesc
  FROM CourseList
  ORDER by Dept_ID ASC, CorNumber ASC
</cfquery>

<h3>cfgrid Example</h3>
<i>Try adding a course to the database, and then deleting it.</i>

<cfform>
  <cfgrid name = "FirstGrid" width = "450"
    query = "GetCourses" insert = "Yes" delete = "Yes"
    font = "Tahoma" rowHeaders = "No"
    colHeaderBold = "Yes"
    selectMode = "EDIT"
    insertButton = "Insert a Row" deleteButton = "Delete selected row">
  </cfgrid>
  <cfinput type="submit" name="gridEntered">
</cfform>...
**cfheader**

**Description**

Generates custom HTTP response headers to return to the client.

**Category**

Data output tags, Page processing tags

**Syntax**

```xml
<cfheader
    charset="character set"
    name = "header name"
    value = "header value">

OR

<cfheader
    statusCode = "status code"
    statusText = "status text">
```

⚠️ **Note**

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

**See also**

cfcache,cfflush,cfhtmlhead,cfinclude,cfsetting,cfsilent,cfcontent

**History**

ColdFusion MX 6.1: Changed behavior for the name attribute: **cfheader name="Content-Disposition"** uses the default file character encoding to encode this header’s value, so the name of a file can include characters in the character encoding used in the file.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>charset</td>
<td>Optional</td>
<td>UTF-8</td>
<td>The character encoding in which to encode the header value. The following list includes commonly used values:</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• utf-8&lt;br&gt;• iso-8859-1&lt;br&gt;• windows-1252&lt;br&gt;• us-ascii&lt;br&gt;• shift_jis&lt;br&gt;• iso-2022-jp&lt;br&gt;• euc-jp&lt;br&gt;• euc-kr&lt;br&gt;• big5&lt;br&gt;• euc-cn&lt;br&gt;• utf-16&lt;br&gt;For more information about character encodings, see <a href="http://www.w3.%5C_org/International/O-charset.html">www.w3.\_org/International/O-charset.html</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>Required if statusCode not specified</th>
<th>Header name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>statusCode</td>
<td>Required if name not specified</td>
<td>Number. HTTP status code.</td>
</tr>
<tr>
<td>statusText</td>
<td>Optional</td>
<td>Explains the status code.</td>
</tr>
<tr>
<td>value</td>
<td>Optional</td>
<td>HTTP header value.</td>
</tr>
</tbody>
</table>

**Usage**

If you use this tag after the `cfflush` tag on a page, an error is thrown.

**Example**

```
<h3>cfheader Example</h3>
<p>cfheader generates custom HTTP response headers to return to the client.</p>
<p>This example forces browser client to purge its cache of requested file.  
<cfheader name="Expires" value="#GetHttpTimeString(Now())#"/>
```
cfhtmlhead

Description

Writes text to the head section of a generated HTML page.

Category

Page processing tags

Syntax

```xml
<cfhtmlhead
text = "{{text}}">
```

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfcache,cfflush,cfheader,cfinclude,cfsetting,cfsilent

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Required</td>
<td></td>
<td>Text to add to the &lt;head&gt; area of an HTML page.</td>
</tr>
</tbody>
</table>

Usage

Use this tag for embedding JavaScript code, or putting other HTML tags, such as meta, link, title, or base in an HTML page header. If you use this tag after the cfflush tag on a page, an error is thrown.

Example

```html
<!--- This example adds a favicon to the HTML Head of every page view. The extra CRLF at the end cleans up view source. Note the embedded tag uses double quotes and the cfhtmlhead text attribute uses single quotes, but it could be the other way around too. --->

<cfhtmlhead
text='&lt;link href="/blog/custom/img/favicon.ico" rel="shortcut icon" type="image/x-icon">#chr(13)#chr(10)'
`
<cfhtmltopdf> creates high quality PDF output from a text block containing CFML and HTML using the PDF Service Manager.

The WebKit implementation for <cfhtmltopdf> is referred as PDFG (acronym of PDF Generator) which is shipped as part of the Jetty installer. The component running inside the Jetty server is referred as the service manager, which takes request for PDF conversion from one or many ColdFusion server/s. (In Windows, the service runs as the ColdFusion 11 Add-on service, which processes both Solr and PDFG requests.)

The WebKit implementation for <cfhtmltopdf>:

- Encapsulates the PDF conversion library.
- Performs HTML to PDF conversion in process.

When you generate a PDF using HTML or CFML, you have to ensure that XSS vulnerabilities cannot be exploited. Hence, you must take relevant measures to protect your system against such vulnerabilities. Also, note that ColdFusion provides encoding functions for XSS protection. See the following functions:

- EncodeForHTML
- Canonicalize

**Category**

Data output tags

**See Also**

cfhtmltopdfitem

**Syntax**
```xml
<cfhtmltopdf
  encryption = "AES_128|RC4_40|RC4_128|RC4_128M|None"
  source = "URL|pathname absolute or relative to web root"
  destination = "filename"
  marginBottom = "number"
  marginLeft = "number"
  marginRight = "number"
  marginTop = "number"
  name = "output variable name"
  orientation = "portrait|landscape"
  overwrite = "yes|no"
  ownerPassword = "password"
  pageHeight = "page height in inches(default)"
  pageType = "page type"
  pageWidth = "page width in inches(default)"
  permissions = "permission list"
  saveAsName = "PDF filename"
  unit = "in|cm"
  userPassword = "password"

  HTML and CFML code
</cfhtmltopdf>
```

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| encryption | Optional | none | AES_128
RC4_40
RC4_128
RC4_128M
None |
<p>| destination | Optional | | Pathname of a file to contain the PDF output. If you omit the destination attribute, ColdFusion displays the output in the browser. |
| <strong>marginBottom</strong> | Optional | Bottom margin in inches (default) or centimeters. To specify the bottom margin in centimeters, include the <code>unit=cm</code> attribute. |
| <strong>marginLeft</strong>  | Optional  | Left margin in inches (default) or centimeters. To specify the left margin in centimeters, include the <code>unit=cm</code> attribute. |
| <strong>marginRight</strong> | Optional  | Right margin in inches (default) or centimeters. To specify the right margin in centimeters, include the <code>unit=cm</code> attribute. |
| <strong>marginTop</strong>   | Optional  | Top margin in inches (default) or centimeters. To specify the top margin in centimeters, include the <code>unit=cm</code> attribute. |
| <strong>name</strong>        | Optional  | Name of an existing variable into which the tag stores the PDF. |
| <strong>orientation</strong> | Optional  | Page orientation: portrait, landscape |
| <strong>overwrite</strong>   | Optional  | no | Specifies whether ColdFusion overwrites an existing file. Used in conjunction with the <code>destination</code> attribute. |
| <strong>ownerPassword</strong> | Optional | Specifies the owner password. Cannot be same as <code>userPassword</code>. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Optional</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageHeight</td>
<td></td>
<td></td>
<td>Page height in inches (default) or centimeters. This attribute is only valid if pagetype=custom. To specify page height in centimeters, include the unit=cm attribute.</td>
</tr>
<tr>
<td>pageType</td>
<td></td>
<td>letter</td>
<td>Page type into which ColdFusion generates the report:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>legal: 8.5 inches x 14 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>letter: 8.5 inches x 11 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A4: 8.27 inches x 11.69 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A5: 5.81 inches x 8.25 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B4: 9.88 inches x 13.88 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B5: 7 inches x 9.88 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B4-JIS: 10.13 inches x 14.31 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B5-JIS: 7.19 inches x 10.13 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>custom</td>
<td>custom: custom height and width. If you specify custom, also specify the pageHeight and pageWidth attributes, and optionally specify the margin attributes and whether the units are inches or centimeters.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Optional</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>pageWidth</td>
<td></td>
<td>Page width in inches (default) or centimeters. This attribute is only valid if pageType=custom. To specify page width in centimeters, include the unit=cm attribute.</td>
<td></td>
</tr>
<tr>
<td>permissions</td>
<td></td>
<td>(format=&quot;PDF&quot; only) Sets one or more of the following permissions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AllowPrinting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AllowModifyContents</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AllowCopy</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AllowModifyAnnotations</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AllowFillIn</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AllowScreenReaders</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AllowAssembly</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AllowDegradedPrinting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AllowSecure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>All</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separate multiple permissions with commas.</td>
<td></td>
</tr>
<tr>
<td>saveAsName</td>
<td></td>
<td>The filename that appears in the SaveAs dialog when a user saves a PDF file written to the browser.</td>
<td></td>
</tr>
<tr>
<td>unit</td>
<td></td>
<td>Optional in: inches. cm: centimeters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default unit for the pageHeight, pageWidth, and margin attributes.</td>
<td></td>
</tr>
</tbody>
</table>
userPassword | Optional | Specifies a user password. Cannot be same as ownerPassword.

**Note:** `<cfhtmltopdfltem>` is added to support adding header/footer/pagebreak in the generated PDF. See The new `<cfhtmltopdfltem>` tag.

**Examples**

The following example shows the most basic usage of `<cfhtmltopdf>`, to create a PDF from CFML code, returning a PDF for display:

```cfml
<cfhtmltopdf>
  This is a test <cfoutput>#now()#</cfoutput>
</cfhtmltopdf>
```

The following example shows the most basic usage of `<cfhtmltopdf>`, to create a PDF from content returned from a URL, returning a PDF for display:

```cfml
<cfhtmltopdf source="http://www.google.com/" />
```

The following example shows using options to control height, width, and save the content to a file instead of for display. The file is saved (by default) into the same directory as the template containing the code:

```cfml
<cfhtmltopdf
  destination="usage_example.pdf" overwrite="yes"
  source="http://www.google.com/"
  unit="in" pageheight="8" pagewidth="4"
  pagetype="custom" />
```

The following example shows how you can set margins, and also adds code to display the resulting file to the user using CFContent:
The following example shows how you can protect the PDF, requiring the user to enter a password to open the file:

```cfc
cfhtmltopdf destination="usage_example3.pdf" 
source="http://www.google.com" overwrite="true"
orientation="portrait" pagetype="A4" margintop="1" marginbottom="1"
marginleft="1" marginright="1" ownerpassword="owner" userpassword="user"
encryption="RC4_128" permissions="AllowPrinting,AllowCopy" />
```

© 2014 Adobe Systems Incorporated. All rights reserved.
<cfhtmltopdfitem>
  type = header/footer/pagebreak
  isBase64 = "yes|no"
  showonprint = "yes|no"
  align = "left|right|center"
  marginleft = "number"
  marginright = "number"
  margintop = "number"
  marginbottom = "number"
  opacity = "number"
  numberformat = "LOWERCASEROMAN|NUMERIC|UPPERCASEROMAN"
  image = " URL|pathname absolute or relative to web root"
</cfhtmltopdfitem>

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

<cfhtmltopdfitem> specifies the action items for a PDF document created by the <cfhtmltopdf> tag. The type attribute takes the following values:

- header
- footer
- pagebreak

**Category**

Data output tags

**Syntax**

```xml
<cfhtmltopdfitem
type = header/footer/pagebreak
isBase64 = "yes|no"
showonprint = "yes|no"
align = "left|right|center"
marginleft = "number"
marginright = "number"
margintop = "number"
marginbottom = "number"
opacity = "number"
numberformat = "LOWERCASEROMAN|NUMERIC|UPPERCASEROMAN"
image = " URL|pathname absolute or relative to web root"
</cfhtmltopdfitem>
```
<table>
<thead>
<tr>
<th><strong>type</strong></th>
<th><strong>Required</strong></th>
<th><strong>none</strong></th>
<th>Specifies the action:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>• header: uses the text/image specified between the <code>&lt;cfhtmltopdfitem&gt;</code> and <code>&lt;/cfhtmltopdfitem&gt;</code> tags as the running header.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• footer: uses the text/image between the <code>&lt;cfhtmltopdfitem&gt;</code> and <code>&lt;/cfhtmltopdfitem&gt;</code> tags as the running footer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• pagebreak: can be used to insert a pagebreak in the generated PDF. When <code>&lt;cfhtmltopdf&gt;</code> is used as a service, pagebreak will not work.</td>
</tr>
</tbody>
</table>

**isBase64**

Optional

Yes/No. Use this only when the image attribute is given a base64 image string.
<table>
<thead>
<tr>
<th>attribute</th>
<th>Optional</th>
<th>value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>showonprint</td>
<td>Optional</td>
<td></td>
<td>Show or hide header/footer when the document is printed.</td>
</tr>
<tr>
<td>align</td>
<td>Optional</td>
<td>Center</td>
<td>Alignment of the text. Left, Right, or Center.</td>
</tr>
<tr>
<td>marginleft</td>
<td>Optional</td>
<td>1.0</td>
<td>Left margin in inches (default) or centimeters. To specify the left margin in centimeters, include the unit=cm attribute.</td>
</tr>
<tr>
<td>marginright</td>
<td>Optional</td>
<td>1.0</td>
<td>Right margin in inches (default) or centimeters. To specify the left margin in centimeters, include the unit=cm attribute.</td>
</tr>
<tr>
<td>margintop</td>
<td>Optional</td>
<td>0.5</td>
<td>Top margin in inches (default) or centimeters. To specify the left margin in centimeters, include the unit=cm attribute.</td>
</tr>
<tr>
<td>marginbottom</td>
<td>Optional</td>
<td>0.5</td>
<td>Bottom margin in inches (default) or centimeters. To specify the bottom margin in centimeters, include the unit=cm attribute.</td>
</tr>
<tr>
<td>opacity</td>
<td>Optional</td>
<td>10</td>
<td>Opacity of the header/footer. Specify a valid number. A number between 1 and 10.</td>
</tr>
<tr>
<td>numberformat</td>
<td>Optional</td>
<td>Numeric</td>
<td>The page number format to be used. LOWERCASEROMAN</td>
</tr>
</tbody>
</table>
**image**  
Optional  
The image file name or object to be used as header or footer.

The following parameters are supported:
- A path to the image file
- A Base-64 string
- A byte array
- A `<cfimage>` object

**Examples**

```cfml
cfhtmltopdf destination="myPDF.pdf" source="http://somesite.com" overwrite="true">
  <cfhtmltopdfitem type="header">
    Page: _PAGENUMBER of _LASTPAGENUMBER
  </cfhtmltopdfitem>
  <cfhtmltopdfitem type="pagebreak" />
  <cfhtmltopdfitem type="footer" image="test.jpg">
  </cfhtmltopdfitem>
</cfhtmltopdf>
```

You can access the following scope variables in `<cfhtmltopdfitem>` content:

- `_PAGENUMBER` – Add current page number.
- `_LASTPAGENUMBER` – Add last page number.
**cfhttp**

**Description**

Generates an HTTP request and handles the response from the server.

**Category**

Internet protocol tags

**Syntax**

```xml
<cfhttp
  url = "server URL"
  charset = "character encoding"
  clientCert = "filename"
  clientCertPassword = "password"
  columns = "query columns"
  delimiter = "character"
  file = "filename"
  firstrowasheaders = "yes|no"
  getAsBinary = "auto|yes|no|never"
  method = "method name"
  multipart = "yes|no"
  name = "query name"
  password = "password"
  path = "path"
  port = "port number"
  proxyServer = "host name"
  proxyPort = "port number"
  proxyUser = "username"
  proxyPassword = "password"
  redirect = "yes|no"
  resolveURL = "yes|no"
  result = "result name"
  textQualifier = "character"
  throwOnError = "yes|no"
  timeout = "time-out period in seconds"
  username = "username"
  userAgent = "user agent"
  authType = "BASIC|NTLM"
  domain = "domain name"
  workstation = "workstation name">

  cfhttpparam tags [optional for some methods]

</cfhttp>
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

© 2014 Adobe Systems Incorporated. All rights reserved.
cfhtpparam, GetHttpRequestData, cfftp, cfldap, cfmail, cfpop

History

ColdFusion 11: Added support for NTLM authentication. Three attributes added - authType, domain, and workstation. Also, this tag supports SNI (server name indication) only if ColdFusion is running on an Oracle 1.7+ JVM.

ColdFusion 8: Added the clientCert and clientCertPassword attributes.

ColdFusion MX 7.01: Added the "never" value of thegetAsBinary attribute.

ColdFusion MX 7: Added the result attribute.

ColdFusion MX 6.1:

- Added support for the following methods: HEAD, PUT, DELETE, OPTIONS, TRACE.
- Added multipart, getAsBinary, proxyUser, and proxyPassword attributes.
- Changed httpparam behavior: all operations can have httpparam tags.
- Added the cfhttp.errorDetail return variable.
- Modified response body content types considered to be text.
- Changed behavior for multiple headers: multiple headers of the same type are returned in an array.
- Added support for HTTPS proxy tunneling.
- Fixed bugs in code and documentation.

ColdFusion MX:

- Added the charset and firstrowasheaders attributes.
- Changed Secure Sockets Layer (SSL) support: ColdFusion uses the Sun JSSE library, which supports 128-bit encryption, to support SSL.

Attributes

The following attributes control the HTTP transaction and can be used for all HTTP methods:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Required</td>
<td>Uses the http protocol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address of the resource on the server that handles the request. The URL must include the hostname or IP address. If you do not specify the transaction protocol (http:// or https://), ColdFusion uses the default protocol, http. If you specify a port number in this attribute, it overrides any port attribute value. The cfhtpparam tag URL attribute appends query string attribute-value pairs to the URL.</td>
</tr>
<tr>
<td>charset</td>
<td>Optional</td>
<td>For request: UTF-8 For response: charset specified by response Content-Type header, or UTF-8 if response does not specify charset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The character encoding of the request, including the URL query string and form or file data, and the response. The following list includes commonly used values:</td>
</tr>
</tbody>
</table>
|             |                   | • utf-8  
|             |                   | • iso-8859-1  
|             |                   | • windows-1252  
|             |                   | • us-ascii  
|             |                   | • shift_jis  
|             |                   | • iso-2022-jp  
|             |                   | • euc-jp  
|             |                   | • euc-kr  
|             |                   | • big5  
|             |                   | • euc-cn  
|             |                   | • utf-16  
<p>|             |                   | For more information character encodings, see <a href="http://www.w3.org/International/O-charset.html">www.w3.org/International/O-charset.html</a>.                                                               |
| clientCert  | Optional          | The full path to a PKCS12 format file that contains the client certificate for the request.                                                                                                                |</p>
<table>
<thead>
<tr>
<th>clientCertPassword</th>
<th>Optional</th>
<th>Password used to decrypt the client certificate.</th>
</tr>
</thead>
<tbody>
<tr>
<td>compression</td>
<td>Optional</td>
<td>The target webserver's compression status. The only supported value is none. If the target website runs on IIS with HTTP compression enabled, use this attribute to avoid a connection failure while performing GET or POST operations.</td>
</tr>
</tbody>
</table>
| getAsBinary | Optional | no | • no: if ColdFusion does not recognize the response body type as text, converts it to a ColdFusion object.
• auto: if ColdFusion does not recognize the response body type as text, converts it to ColdFusion Binary type data.
• yes: always converts the response body content into ColdFusion Binary type data, even if ColdFusion recognizes the response body type as text.
• never: prevents the automatic conversion of certain MIME types to the ColdFusion Binary type data; treats the returned content as text.

ColdFusion recognizes the response body as text if:
• the header does not specify a content type.
• the content type starts with "text".
• the content type starts with "message".
• the content type is "application/octet-stream".

If ColdFusion does not recognize the body as text and converts it to an object, but the body consists of text, the `cfo utput` tag can display it. The `cfoutput` tag cannot display Binary type data. (To convert binary data to text, use the `ToS tring` function.) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>Optional</td>
<td>GET</td>
<td></td>
</tr>
</tbody>
</table>
- **GET**: requests information from the server. Any data that the server requires to identify the requested information must be in the URL or in `cfhttp` tags.
- **POST**: sends information to the server for processing. Requires one or more `cfhttpparam` tags. Often used for submitting form-like data.
- **PUT**: requests the server to store the message body at the specified URL. Use this method to send files to the server.
- **DELETE**: requests the server to delete the specified URL.
- **HEAD**: identical to the `GET` method, but the server does not send a message body in the response. Use this method for testing hypertext links for validity and accessibility, determining the type or modification time of a document, or determining the type of server.
**TRACE**: requests that the server echo the received HTTP headers back to the sender in the response body. Trace requests cannot have bodies. This method enables the ColdFusion application to see what is being received at the server, and use that data for testing or diagnostic information.

**OPTIONS**: a request for information about the communication options available for the server or the specified URL. This method enables the ColdFusion application to determine the options and requirements associated with a URL, or the capabilities of a server, without requesting any additional activity by the server.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>Optional</td>
<td>Use to pass a password to the target URL for Basic Authentication. Combined with <code>username</code> to form a base64 encoded string that is passed in the <code>Authenticate</code> header. Does not provide support for Integrated Windows, NTLM, or Kerberos authentication.</td>
</tr>
<tr>
<td>port</td>
<td>Optional</td>
<td>80 for http, 443 for https</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Default</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>proxyServer</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>proxyPort</td>
<td>Optional</td>
<td>80</td>
</tr>
<tr>
<td>proxyUser</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>proxyPassword</td>
<td>Optional</td>
<td></td>
</tr>
</tbody>
</table>
| redirect     | Optional | yes     | If the response header includes a Location field AND ColdFusion receives a 300-series (redirection) status code, specifies whether to redirect execution to the URL specified in the field:  
  * yes: redirects execution to the specified page.  
  * no: stops execution and returns the response information in the cfhttp variable, or throws an error if the throwOnError attribute is True. The cfhttp.responseHeader.Location variable contains the redirection path. ColdFusion follows a maximum of four redirects on a request. If there are more, ColdFusion functions as if redirect = "no".  

Note: The cflocation tag generates an HTTP 302 response with the url attribute as the Location header value.
<table>
<thead>
<tr>
<th><strong>.resolveURL</strong></th>
<th><strong>Optional</strong></th>
<th><strong>no</strong></th>
</tr>
</thead>
</table>
|                |             | • no: does not resolve URLs in the response body. As a result, any relative URL links in the response body do not work.  
|                |             | • yes: resolves URLs in the response body to absolute URLs, including the port number, so that links in a retrieved page remain functional. Applies to these HTML tags:  
|                |             | • img  
|                |             | • src  
|                |             | • a href  
|                |             | • form action  
|                |             | • applet code  
|                |             | • script src  
|                |             | • embed src  
|                |             | • embed pluginspace  
|                |             | • body background  
|                |             | • frame src  
|                |             | • bgsound src  
|                |             | • object data  
|                |             | • object classid  
|                |             | • object codebase  
|                |             | • object usemap  
<p>|                |             | Does not resolve URLs if the file and path attributes are used.  |
| <strong>result</strong>     | <strong>Optional</strong> | <strong>Lets you specify an alternate variable in which to receive a result.</strong> |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| throwOnError | Optional | no     | • yes: if the server returns an error response code, throws an exception that can be caught using the cftry and cfcatch or ColdFusion error pages.  
  • no: does not throw an exception if an error response is returned. In this case, your application can use the cfhttp.StatusCode variable to determine if there was an error and its cause. |
<p>| timeout      | Optional |        | Value, in seconds, that is the maximum time the request can take. If the time-out passes without a response, ColdFusion considers the request to have failed. If the client specifies a time-out in the URL search parameter (for example, ?RequestTime=120) ColdFusion uses the lesser of the URL time-out and the timeout attribute value; this ensures that the request times out before, or at the same time as, the page. If the URL does not specify a time-out, ColdFusion uses the lesser of the Administrator time-out and the timeout attribute value. If the time-out is not set in any of these, ColdFusion waits indefinitely for the cfhttp request to process. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userAgent</td>
<td>Optional</td>
<td>ColdFusion</td>
<td>Text to put in the user agent request header. Used to identify the request client software. Can make the ColdFusion application appear to be a browser.</td>
</tr>
<tr>
<td>username</td>
<td>Optional</td>
<td></td>
<td>Use to pass a user name to the target URL for Basic Authentication. Combined with password to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerberos authentication.</td>
</tr>
<tr>
<td>authType</td>
<td>Optional</td>
<td>BASIC</td>
<td>The supported values are: • BASIC • NTLM Note: For NTLM, to work, the redirect attribute must be set to false.</td>
</tr>
<tr>
<td>domain</td>
<td>Optional</td>
<td></td>
<td>The domain name for authentication. (Use for NTLM-based authentication)</td>
</tr>
<tr>
<td>workstation</td>
<td>Optional</td>
<td></td>
<td>The workstation name for authentication. (Use for NTLM-based authentication)</td>
</tr>
</tbody>
</table>

The following attribute is used with the PUT method to determine how to send data specified with `httpparam type="formField"`:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Req/Opt</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>multipartType</td>
<td>Optional</td>
<td>form-data</td>
<td>Allows you to set the multipart header field to <code>related</code> or <code>form-data</code>. By default, the value is <code>form-data</code>.</td>
</tr>
</tbody>
</table>

Example:

```cfttp
<!--- Get Video --->
<cfset videoName = "<vedio path>\hello.wmv">
```
<cfset videoFileName = "hello.wmv">
<cfoutput>
<!--- Set User Account Data --->
<cfset clientKey = "client key from google"/>
<cfset devKey = "<dev key from google>"/>
<!--- Authenticate with Google / YouTube --->
<cfhttp url="https://www.google.com/accounts/ClientLogin" method="post" result="result" charset="utf-8">
   <cfhttpparam type="formfield" name="accountType" value="HOSTED_OR_GOOGLE">
   <cfhttpparam type="formfield" name="Email" value="<gmail id>">
   <cfhttpparam type="formfield" name="Passwd" value="<password>">
   <cfhttpparam type="formfield" name="service" value="youtube">
   <cfhttpparam type="formfield" name="source" value="youtubecode">
</cfhttp>
<!--- Create Auth Token --->
<cfset authdata = structNew()>
<cfloop index="line" list="#content#" delimiters="#chr(10)#">
   <cfset dtype = listFirst(line, ";")>
   <cfset value = listRest(line, ";")>
   <cfset authdata[dtype] = value></cfloop>
<!--- Create ATOM XML and save to a file to be sent with video --->
<cfsavecontent variable="meta"><cfoutput>
   <entry xmlns="http://www.w3.org/2005/Atom"
   xmlns:media="http://search.yahoo.com/mrss/"
   xmlns:yt="http://gdata.youtube.com/schemas/2007">
   <media:group>
   <media:title type="plain">WithOutQuotes</media:title>
   <media:description type="plain">Test Description</media:description>
   <media:category
   <media:keywords>yourvideo</media:keywords>
   </media:group>
</entry>
</cfoutput>
</csavecontent>
<cfset tmpfile = expandPath("./meta.xml")>
<cffile action="write" file="#tmpfile#" output="#trim(meta)#" />

<!--- Upload video --->
<cfhttp url="#YouTubeUploadURL#" result="result" method="POST" timeout="450"
 multipartType="related">
   <cfhttpparam type="header" name="Authorization" value="GoogleLogin
auth=#authdata.auth#">
   <cfhttpparam type="header" name="X-GData-Client" value="#variables.clientkey#">
   <cfhttpparam type="header" name="X-GData-Key" value="#variables.devkey#">
   <cfhttpparam type="header" name="Slug" value="#videoFileName#">
   <!--- Send 2 files --->
   <cfhttpparam type="file" name="API_XML_Request" file="#tmpfile#"
   mimetype="application/atom+xml">
   <cfhttpparam type="file" name="file" file="#videoName#" mimetype="video/*">
</cfhttp>
The following attribute allows you to specify the name of the variable in which you would like the results of the operation returned. The name you specify replaces `cfhttp` as the prefix by which you access the returned variables. For example, if you set the `result` attribute to `myResult`, you would access `FileContent` as `myResult.FileContent#`.

The `result` attribute allows functions or CFCs that are called from multiple pages at the same time to avoid overwriting the results of one call with another. For information about the variables returned by a `cfhttp` get operation, see the section *Variables returned by a cfhttp get operation* in the Usage section.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>result</code></td>
<td>Optional</td>
<td></td>
<td>Specifies the name of the variable in which you want the result returned.</td>
</tr>
</tbody>
</table>

The following attributes tell ColdFusion to put the HTTP response body in a file. You can put the response body in a file for GET, POST, PUT, DELETE, OPTIONS, and TRACE methods, but it is generally not useful with the DELETE or OPTIONS method.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>file</code></td>
<td>Required if <code>path</code> is specified and not a GET method</td>
<td>See Description</td>
<td>Name of the file in which to store the response body. For a GET operation, the default is the file requested in the URL, if there is one. For example, if the URL in a GET method is <code>http://www.myco.com/test.htm</code>, the default file is <code>test.htm</code>. Do not specify the path to the directory in this attribute; use the <code>path</code> attribute.</td>
</tr>
<tr>
<td><code>path</code></td>
<td>Required if <code>file</code> is specified.</td>
<td></td>
<td>Tells ColdFusion to save the HTTP response body in a file. Contains the absolute path to the directory in which to store the file.</td>
</tr>
</tbody>
</table>

Use the following syntax in the `path` attribute to specify an in-memory directory for your files. In-memory files speed processing of transient data.
The filepath can include multiple directories, for example `ram:///petStore/images`. Create the directories in the path before you can use them. For more information on using in-memory files, see *Working with in-memory files* in Optimizing ColdFusion applications in the Developing ColdFusion Applications.

The following attributes tell ColdFusion to convert the HTTP response body into a ColdFusion query object. They can be used with the GET and POST methods only.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columns</td>
<td>Optional</td>
<td>First row of response contains column names.</td>
<td>The column names for the query, separated by commas, with no spaces. Column names must start with a letter. The remaining characters can be letters, numbers, or underscore characters (_). If there are no column name headers in the response, specify this attribute to identify the column names. If you specify this attribute, and the <code>firstrowasHeader</code> attribute is True (the default), the column names specified by this attribute replace the first line of the response. You can use this behavior to replace the column names retrieved by the request with your own names. If a duplicate column heading is encountered in either this attribute or in the column names from the response, ColdFusion appends an underscore to the name to make it unique. If the number of columns specified by this attribute does not equal the number of columns in the HTTP response body, ColdFusion generates an error.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>delimiter</td>
<td>Optional</td>
<td>, (comma)</td>
<td>A character that separates query columns. The response body must use this character to separate the query columns.</td>
</tr>
</tbody>
</table>
| firstrowasheaders | Optional | yes                  | Determines how ColdFusion processes the first row of the query record set:  
- **yes**: processes the first row as column heads. If you specify a **columns** attribute, ColdFusion ignores the first row of the file.  
- **no**: processes the first row as data. If you do not specify a **columns** attribute, ColdFusion generates column names by appending numbers to the word "column"; for example, "column_1". |
| name          | Optional |                       | Tells ColdFusion to create a query object with the given name from the returned HTTP response body.                                        |
| textQualifier | Optional | " double-quotation mark | A character that, optionally, specifies the start and end of a text column. This character must surround any text fields in the response body that contain the delimiter character as part of the field value.  
To include this character in column text, escape it by using two characters in place of one. For example, if the qualifier is a double-quotation mark, escape it as "". |
The **cfhttp** tag is a general-purpose tool for creating HTTP requests and handling the returned results. It enables you to generate most standard HTTP request types. You use embedded **cfhttpparam** tags to specify request headers and body content.

When ColdFusion receives a response to a **cfhttp** request, it can put the response body (if any) in a file or the **cfhttp.FileContent** string variable. If the body text is structured as a result set, ColdFusion can put the body text in a **cfhttp.query** object. You can also access the values of all returned headers and specify how to handle error status and redirections, and specify a time-out to prevent requests from hanging.

The HTTP protocol is the backbone of the World Wide Web and is used for every web transaction. Because the **cfhttp** tag can generate most types of requests, it provides significant flexibility. Possible uses include:

- Interacting with dynamic web sites and services that are not available as web services. (Use the **cfinvoke** tag to access SOAP web services.)
- Getting the contents of an HTML page or other file such as an image on a web server for use in your CFML page or storage in a file.
- Sending a secure request to a server by specifying the https protocol in the **url** attribute.
- Using the POST method to send a multipart/form-data style post to any URL that can handle such data and return results, including CGI executables or even other ColdFusion pages.
- Using the PUT method to upload files to a server that does not accept FTP requests.

This tag can, and for PUT and POST requests must, have a body that contains **cfhttpparam** tags. If this tag has **cfhttpparam** tags, it must have a **</cfhttp>** end tag.

To use HTTPS with the **cfhttp** tag, you might need to manually import the certificate for each web server into the keystore for the JRE that ColdFusion uses. This procedure should not be necessary if the certificate is signed (issued) by an authority that the JSSE (Java Secure Sockets Extension) recognizes (for example, Verisign); that is, if the signing authority is in the cacerts already. However, you might need to use the procedure if you are issuing SSL (secure sockets layer) certificates yourself.

### Manually import a certificate

1. Go to a page on the SSL server in question.
2. Double-click the lock icon.
3. Click the Details tab.
4. Click Copy To File.
5. Select the base64 option and save the file.
6. Copy the CER file into C:\ColdFusion9\runtime\jre\lib\security (or whichever JRE ColdFusion is using).
7. Run the following command in the same directory (keytool.exe is located in C:\ColdFusion9\runtime\jre\bin):

   ```
   keytool -import -keystore cacerts -alias giveUniqueName -file filename.cer
   ```

### Variables returned by a **cfhttp** get operation

The **cfhttp** tag returns the following variables. If you set the **result** attribute, the name you assign replaces **cfhttp** as the prefix. For additional information, see the **result** attribute.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cfhttp.charSet</strong></td>
<td>Response character character set (character encoding) specified by the response Content-Type header.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cfhttp.errorDetail</td>
<td>If the connection to the HTTP server fails, contains details about the failure. For instance: &quot;Unknown host: my.co.com&quot;; otherwise, the empty string. recommends that you check this variable for an error condition before checking other variables.</td>
</tr>
<tr>
<td>cfhttp.fileContent</td>
<td>Response body; for example, the contents of an html page retrieved by a GET operation. Empty if you save the response in a file.</td>
</tr>
<tr>
<td>cfhttp.header</td>
<td>Raw response header containing all header information in a single string. Contains the same information as the cfhttp.responseHeader variable.</td>
</tr>
<tr>
<td>cfhttp.mimeType</td>
<td>MIME type specified by the response Content-Type header; for example, text/html.</td>
</tr>
<tr>
<td>cfhttp.responseHeader</td>
<td>The response headers formatted into a structure. Each element key is the header name, such as Content-Type or Status_Code. If there is more than one instance of a header type, the type values are put in an array. One common technique is to dynamically access the cfhttp.responseHeader structure as a dynamic array; for example, #cfhttp.responseHeaderVaria ble#.</td>
</tr>
<tr>
<td>cfhttp.statusCode</td>
<td>The HTTP status_code header value followed by the HTTP Explanation header value; for example, &quot;200 OK&quot;.</td>
</tr>
<tr>
<td>cfhttp.text</td>
<td>Boolean; true if the response body content type is text. ColdFusion recognizes the response body as text in the following situations:</td>
</tr>
<tr>
<td></td>
<td>• if the header does not specify a content type</td>
</tr>
<tr>
<td></td>
<td>• if the content type starts with &quot;text&quot;</td>
</tr>
<tr>
<td></td>
<td>• if the content type starts with &quot;message&quot;</td>
</tr>
<tr>
<td></td>
<td>• if the content type is &quot;application/octet-stream&quot;</td>
</tr>
</tbody>
</table>

### Building a query from a delimited text file

The cfhttp tag can create a ColdFusion query object form the response body. To do so, the response body must consist of lines of text, with each line having fields that are delimited by a character that identifies the column breaks. The default delimiter is a comma (,). The response data can also use a text qualifier; the default is a double-quotation mark ("). If you surround a string field in the text qualifier, the field can contain the delimiter character. To include the text qualifier in field text, escape it by using a double character. The following line shows a two-line request body that is converted into a query. It has three comma-delimited fields:
Field1,Field2,Field3
"A comma, in text","A quote: "Oh My!"",Plain text

Run the following code to show how ColdFusion treats this data:

```cfhttp
<cfhttp method="Get"
  url="127.0.0.1:8500/tests/escapetest.txt"
  name="onerow">
  <cfdump var="#onerow#"><br>
</cfhttp>
```

Column names can be specified in three ways:

- By default, ColdFusion uses the first row of the response as the column names.
- If you specify a comma-delimited columns attribute, ColdFusion uses the names specified in the attribute as the column names. Set `firstRowAsHeaders="no"` if the first row of the response contains data. Otherwise, ColdFusion ignores the first row.
- If you do not specify a columns attribute and set `firstRowAsHeaders="no"`, ColdFusion generates column names of the form Column_1, Column2, etc. The `cfhttp` tag checks to ensure that column names in the data returned by the tag start with a letter and contain only letters, numbers, and underscore characters (_). ColdFusion checks for invalid column names. Column names must start with a letter. The remaining characters can be letters, numbers, or underscores (_). If a column name is not valid, ColdFusion generates an error.

Notes

- For the ColdFusion Administrator time-out and the URL time-out to take effect, enable the time-out in the ColdFusion Administrator, Server Settings page. For more information, see Configuring and Administering ColdFusion.
- The `cfhttp` tag supports Basic Authentication for all operations.
- The `cfhttp` tag uses SSL to negotiate secure transactions.
- If you put the HTTP response body in a file, ColdFusion does not put it in the CFHTTP.FileContent variable or generate a query object. If you do not put the response body in a file, ColdFusion puts it in the CFHTTP.FileContent variable; if you specify a name attribute ColdFusion generates a query object.
- The `cfhttp` tag does not support NTLM or Digest Authentication.
- If you are using Microsoft IIS, there is no HTTP header size limit. To specify an HTTP header size limit, set it in IIS.

Example
<!--- This example displays the information provided by the Designer & Developer Center XML feed, http://www.adobe.com/devnet/resources/_resources.xml. See http://www.adobe.com/devnet/articles/xml_resource_feed.html for more information on this feed. --->

<!--- Set the URL address. --->
<cfset urlAddress="http://www.adobe.com/devnet/resources/_resources.xml"></cfset>

<!--- Use the CFHTTP tag to get the file content represented by urladdress. Note that />, not an end tag, terminates this tag. --->
<cfhttp url="#urladdress#" method="GET" resolveurl="Yes" throwOnError="Yes"/>

<!--- Parse the XML and output a list of resources. --->
<cfset xmlDoc = XmlParse(CFHTTP.FileContent)>

<!--- Get the array of resource elements, the xmlChildren of the xmlroot. --->
<cfset resources=xmlDoc.xmlroot.xmlChildren>
<cfset numresources=ArrayLen(resources)>

<cfloop index="i" from="1" to="#numresources#">
    <cfset item=resources[i]>
    <cfoutput>
        <strong><a href="#item.url.xmltext#">#item.title.xmltext#</a></strong><br>
        <strong>Author</strong> &nbsp;&nbsp;#item.author.xmltext#<br>
        <strong>Applies to these products</strong><br>
        <cfloop index="i" from="4" to="#arraylen(item.xmlChildren)#">
            #item.xmlChildren[i].xmlAttributes.Name#<br>
        </cfloop>
    </cfoutput>
</cfloop>
### cfhttpparam

**Description**

Allowed inside `cfhttp` tag bodies only. Required for `cfhttp` POST operations. Optional for all others. Specifies parameters to build an HTTP request.

**Category**

Internet protocol tags

**Syntax**

```
<cfhttpparam
  type = "transaction type"
  encoded = "yes|no"
  file = "filename"
  mimeType = "MIME type designator"
  name = "data name"
  value = "data value">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

`cfhttp`, `GetHttpRequestData`, `cfftp`, `cfldap`, `cfmail`, `cfmailparam`, `cfpop`

**History**

ColdFusion MX 6.1:

- Added the `header` and `body` types.
- Added the `encoded` and `mimeType` attributes.
- Changed HTTP method behavior: all HTTP methods can have `httpparam` tags.
- Changed the `name` attribute requirements: it is not required for all types.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Required</td>
<td></td>
<td>Information type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <code>header</code>: specifies an HTTP header. ColdFusion does not URL encode the header.</td>
</tr>
</tbody>
</table>
• CGI: specifies an HTTP header. ColdFusion URL encodes the header by default.
• body: specifies the body of the HTTP request. ColdFusion does not automatically set a content-type header or URL encode the body contents. To specify the content-type, use a separate cfhttpparam tag with type=header.
• XML: identifies the request as having a content-type of text/xml. Specifies that the value attribute contains the body of the HTTP request. Used to send XML to the destination URL. ColdFusion does not URL encode the XML data.
• file: tells ColdFusion to send the contents of the specified file. ColdFusion does not URL encode the file contents.
• URL: specifies a URL query string name-value pair to append to the cfhttpurl attribute. ColdFusion URL encodes the query string.
• formField: specifies a form field to send. ColdFusion URL encodes the Form field by default.
• cookie: specifies a cookie to send as an HTTP header. ColdFusion URL encodes the cookie.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>encoded</td>
<td>Optional</td>
<td>yes</td>
<td>Applies to FormField and CGI types; ignored for all other types. Specifies whether to URL encode the form field or header.</td>
</tr>
<tr>
<td>file</td>
<td>Required only if type=&quot;File&quot;</td>
<td>Applies to File type; ignored for all other types. The absolute path to the file that is sent in the request body.</td>
<td></td>
</tr>
<tr>
<td>mimeType</td>
<td>Optional</td>
<td>Applies to File type; invalid for all other types. Specifies the MIME media type of the file contents. The content type can include an identifier for the character encoding of the file; for example, text/html; charset=ISO-8859-1 indicates that the file is HTML text in the ISO Latin-1 character encoding.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Required. Optional (and ignored) for Body and XML types</td>
<td>Variable name for data that is passed. Ignored for Body and XML types. For File type, specifies the filename to send in the request.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>Required. Optional (and ignored) for File type</td>
<td>Value of the data that is sent. Ignored for File type. The value must contain string data or data that ColdFusion can convert to a string for all type attributes except Body. Body types can have string or binary values.</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

Specifies header or body data to send in the HTTP request. The `type` attribute identifies the information that the parameter specifies. A `cfhttp` tag can have multiple `cfhttpparam` tags, subject to the following limitations:

- An XML `type` attribute cannot be used with additional XML `type` attributes, or with body, file, or formField `type` attributes.
- A body `type` attribute cannot be used with additional body `type` attributes, or with XML, file, or formField `type` attributes.
The XML and body type attributes cannot be used with the cfhttp tag TRACE method.

- The file type attribute is only meaningful** with the cfhttp tag POST and PUT methods.
- The formField type attribute is only meaningful** with the cfhttp tag POST and GET methods.

If you send an HTTP request to a ColdFusion page, all HTTP headers, not just those sent using the CGI type, are available as CGI scope variables. However, any custom variables (such as “myVar”) do not appear in a dump of the CGI scope.

When you send a file using the type="file" attribute, the file content is sent in the body of a multipart/form-data request. If you send the file to a ColdFusion page, the Form scope of the receiving page contains an entry with the name you specified in the cfhttpparam tag name attribute as the key. The value of this variable is the path to a temporary file containing the file that you sent. If you also send Form field data, the location of the filename in the form.fieldnames key list depends on the position of the cfhttpparam tag with the file relative to the cfhttp tags with the form data. URL-encoding preserves special characters (such as the ampersand) when they are passed to the server. For more information, see the function URLEncoded Format. To send arbitrary data in a "raw" HTTP message, use a cfhttpparam tag with a type="body" attribute to specify the body content and use cfhttpparam tags with a type="header" attributes to specify the headers.

Example
<!--- This example consists of two CFML pages. 
The first page posts to the second. --->

<!--- The first, posting page. 
This page posts variables to another page and displays the body of the response from 
the second page. Change the URL and port as necessary for your environment. --->

<cfhttp
    method="post"
    url="http://127.0.0.1/tests/http/cfhttpparamexample.cfm"
    port="8500"
    throwonerror="Yes">
    <cfhttpparam name="form_test" type="FormField" value="This is a form variable">
    <cfhttpparam name="url_test" type="URL" value="This is a URL variable">
    <cfhttpparam name="cgi_test" type="CGI" value="This is a CGI variable">
    <cfhttpparam name="cookie_test" type="Cookie" value="This is a cookie">
</cfhttp>

<!--- Output the results returned by the posted-to page. --->
<cfoutput>
    #cfhttp.fileContent#
</cfoutput>

<!--- This is the cfhttpparamexample.cfm page that receives and processes the Post request. Its response body is the generated HTML output. --->

<h3>Output the passed variables</h3>
<cfoutput>
    Form variable: #form.form_test#
    <br>URL variable: #URL.url_test#
    <br>Cookie variable: #Cookie.cookie_test#
    <br>CGI variable: #CGI.cgi_test#<br>
    <br>Note that the CGI variable is URL encoded.
</cfoutput>
Tags i

This section lists tags starting with alphabet I with their description.

- cfif
- cfimage
- cfimap
- cfimapfilter
- cfimpersonate
- cfimport
- cfinclude
- cfindex
- cfinput
- cfinsert
- cfinterface
- cfinvoke
- cfinvokeargument
**cfif**

**Description**

Creates simple and compound conditional statements in CFML. Tests an expression, variable, function return value, or string. Used, optionally, with the `cfelse` and `cfelseif` tags.

**Category**

*Flow-control tags*

**Syntax**

```
<cfif expression>
    HTML and CFML tags
    <cfelseif expression>
    HTML and CFML tags
    <cfelse>
    HTML and CFML tags
</cfif>
```

**See also**

`cfelse`, `cfelseif`, `cfabort`, `cfbreak`, `cfexecute`, `cfexit`, `cflocation`, `cfloop`, `cfswitch`, `cfthrow`, `cftry`

**Usage**

If the value of the expression in the `cfif` tag is `true`, ColdFusion processes all the code that follows, up to any `cfelseif` or `cfelse` tag, and then skips to the `cfif` end tag. Otherwise, ColdFusion does not process the code that immediately follows the `cfif` tag, and continues processing at any `cfelseif` or `cfelse` tag, or with the code that follows the `cfif` end tag.

When testing the return value of a function that returns a Boolean, you do not have to define the True condition explicitly. This example uses the `IsArray` function:

```
<cfif IsArray(myarray)>
```

If successful, `IsArray` evaluates to yes, the string equivalent of the Boolean True. This is preferred over explicitly defining the True condition this way:

```
<cfif IsArray(myarray) IS True>
```

This tag requires an end tag.

**Example**

In this example, variables are shown within number signs. This is not required.

```
<!--- This example shows the interaction of cfif, cfelse, and cfelseif. --->
```
<!----- First, perform a query to get some data. ----->
<cfquery name="getCenters" datasource="cfdocexamples">
  SELECT Center_ID, Name, Address1, Address2, City, State, Country, PostalCode, Phone, Contact
  FROM Centers
  ORDER by City, State, Name
</cfquery>
<p>CFIF gives us the ability to perform conditional logic based on a condition or set of conditions.</p>
<p>For example, we can output the list of Centers from the snippets datasource by group and only display them <b>IF</b> City = San Diego.</p>
<hr>
<!---- Use CFIF to test a condition when outputting a query. ----->
<p>The following centers are in San Diego:</p>
<cfoutput query="getCenters">
  <cfif Trim(City) is "San Diego">
    <br><b>Name/Address:</b>#Name#, #Address1#, #City#, #State#
    <br><b>Contact:</b> #Contact#
    <br>
  </cfif>
</cfoutput>
<hr>
<p>If we would like more than one condition to be the case, we can ask for a list of the centers in San Diego <b>OR</b> Santa Ana. If the center does not follow this condition, we can use CFELSE to show only the names and cities of the other centers.</p>
<p>Notice how a nested CFIF is used to specify the location of the featured site (Santa Ana or San Diego).</p>
<!----- Use CFIF to specify a conditional choice for multiple options; also note the nested CFIF. --->
<p>Complete information is shown for centers in San Diego or Santa Ana. All other centers are listed in italic:</p>
<cfoutput query="getCenters">
  <cfif Trim(City) is "San Diego" OR Trim(City) is "Santa Ana">
    <h4>Featured Center in <cfif Trim(City) is "San Diego">San Diego</cfif> <cfelse>Santa Ana</cfif></h4>
    <b>Name/Address:</b>#Name#, #Address1#, #City#, #State#
    <br><b>Contact:</b> #Contact#
    <br>
  </cfif>
  <cfelse>
    <br><i>#Name#, #City#</i>
  </cfelse>
</cfoutput>
<hr>
<p>Finally, we can use CFELSEIF to cycle through a number of conditions and produce varying output. Note that you can use CFCASE and CFSWITCH for a more elegant representation of this behavior.</p>
<!----- Use CFIF in conjunction with CFELSEIF to specify more than one branch in a conditional situation. --->
<cfoutput query="getCenters">
  <cfif Trim(City) is "San Diego" OR Trim(City) is "Santa Ana">
    <br><i>#Name#, #City#</i> (this one is in <cfif Trim(City) is "San Diego">San Diego</cfif> <cfelse>Santa Ana</cfif>)
  </cfif>
</cfoutput>
<cfelseif Trim(City) is "San Francisco">
  <br><i>#Name#, #City#</i> (this one is in San Francisco)
</cfelseif>
<cfelseif Trim(City) is "Suisun">
  <br><i>#Name#, #City#</i> (this one is in Suisun)
</cfelseif>
<cfelse>
  <br><b>Not in a city we track</b>
</cfif>
</cfoutput>
**cfimage**

**Description**

Creates a ColdFusion image. You can use the `cfimage` tag to perform common image manipulation operations as a shortcut to `Image` functions. You can use the `cfimage` tag independently or in conjunction with `Image` functions.

**History**

ColdFusion 10: Added the attribute `interpolation` to `{{cfimage action = "resize"}}`

ColdFusion 8: Added this tag.

**Category**

Other tag

**Syntax**

```
Add a border to an image
<cfimage
  required
  action = "border"
  source = "absolute pathname|pathname relative to the web root|URL|#cfimage variable#"
  optional
  color = "hexadecimal value|web color"
  destination = "absolute pathname|pathname relative to the web root"
  isBase64 = "yes|no"
  name = "cfimage variable"
  overwrite = "yes|no"
  thickness = "number of pixels">

Create a CAPTCHA image
<cfimage
  required
  action = "captcha"
  height = "number of pixels"
  text = "text string"
  width = "number of pixels"
  optional
  destination = "absolute pathname|pathname relative to the web root"
  difficulty = "high|medium|low"
  overwrite = "yes|no"
  fonts = "comma-separated list of font names"
  fontSize = "point size">

Convert an image file format
<cfimage
  required
  action = "convert"
  destination = "absolute pathname|pathname relative to the web root"
  source = "absolute pathname|pathname relative to the web root|URL|#cfimage variable#"
  optional
  isBase64 = "yes|no"
  name = "cfimage variable"
```
overwrite = "yes\|no">

Retrieve information about an image
<cfimage
  required
  action = "info"
  source = "absolute pathname\|pathname relative to the web root\|URL\|#cfimage variable#"
  structname = "structure name"
  optional
  isBase64 = "yes\|no">

Read an image into memory
<cfimage
  required
  name = "cfimage variable"
  source = "absolute pathname\|pathname relative to the web root\|URL\|#cfimage variable#"
  optional
  action = "read"
  isBase64 = "yes\|no">

Resize an image
<cfimage
  required
  action = "resize"
  height = "number of pixels\|percent%"
  source = "absolute pathname\|pathname relative to the web root\|URL\|#cfimage variable#"
  width = "number of pixels\|percent%"
  optional
  destination = "absolute pathname\|pathname relative to the web root"
  isBase64 = "yes\|no"
  name = "cfimage variable"
  overwrite = "yes\|no">
  interpolation = "interpolation algorithm"

Rotate an image
<cfimage
  required
  action = "rotate"
  angle = "angle in degrees"
  source = "absolute pathname\|pathname relative to the web root\|URL\|#cfimage variable#"
  optional
  destination = "absolute pathname\|pathname relative to the web root"
  isBase64 = "yes\|no"
  name = "cfimage variable"
  overwrite = "yes\|no">

Write an image to a file
<cfimage
  required
  action = "write"
  destination = "absolute pathname\|pathname relative to the web root"
  source = "absolute or relative pathname\|URL\|#cfimage variable#"
  optional
  isBase64 = "yes\|no"
  overwrite = "yes\|no"
Write an image to the browser
<cfimage
required
action = "writeToBrowser"
source = "absolute pathname|pathname relative to the web root|URL|#cfimage
variable#"
optional
format = "png|jpg|jpeg"
isNew64= "yes|no">

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

ImageAddBorder, ImageCreateCaptcha, ImageInfo, ImageNew, ImageRead, ImageReadBase64, ImageResize, ImageRotate, ImageWrite, ImageWriteBase64, Creating and Manipulating ColdFusion Images in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>N/A</td>
<td>Optional</td>
<td>read</td>
<td>Action to take. Must be one of the following: border</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The default action is read, which you do not need to specify explicitly.</td>
</tr>
<tr>
<td>angle</td>
<td>rotate</td>
<td>Required</td>
<td></td>
<td>Angle in degrees to rotate the image. Specify an integer for the value.</td>
</tr>
<tr>
<td>color</td>
<td>border</td>
<td>Optional</td>
<td>black</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>----------</td>
<td>-------</td>
<td></td>
</tr>
</tbody>
</table>

Border color. Hexadecimal value or supported named color; see the name list in Valid HTML named colors section. For a hexadecimal value, use the form "##xx xxxx" or "xxxxxxx", where x = 0-9 or A-F; use two number signs or none.
<table>
<thead>
<tr>
<th><strong>destination</strong></th>
<th><strong>border captcha convert resize rotate write</strong></th>
<th><strong>Optional (see Description)</strong></th>
<th><strong>Absolute or relative pathname where the image output is written. The image format is determined by the file extension. The <code>convert</code> and <code>write</code> actions require a <code>destination</code> attribute. The <code>border</code>, <code>captcha</code>, <code>resize</code>, and <code>rotate</code> actions require a <code>name</code> attribute or a <code>destination</code> attribute. You can specify both. ColdFusion supports only CAPTCHA images in PNG format. Where the CAPTCHA image is placed depends on the following:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If <code>destination</code> is specified, image is written to the file (based on the absolute or relative pathname you specify).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If <code>name</code> is specified, image is written to the image variable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If neither <code>destination</code> nor <code>name</code> is specified, image is written to the browser.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>difficulty</strong></th>
<th><strong>captcha</strong></th>
<th><strong>Optional</strong></th>
<th><strong>low</strong></th>
<th><strong>Level of complexity of the CAPTCHA text. Specify one of the following levels of text distortion:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- medium</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- high</td>
</tr>
<tr>
<td>fonts</td>
<td>captcha</td>
<td>Optional</td>
<td>One or more valid fonts to use for the CAPTCHA text. Separate multiple fonts with commas. If the specified font is not available, falls back to system fonts.</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>fontSize</td>
<td>captcha</td>
<td>Optional 24</td>
<td>Font size of the text in the CAPTCHA image. The value must be an integer.</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>writeToBrowser</td>
<td>Optional PNG</td>
<td>Format of the image displayed in the browser. If you do not specify a format, the image is displayed in PNG format. You cannot display a GIF image in a browser. GIF images are displayed in PNG format.</td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>captcha resize</td>
<td>Required</td>
<td>Height in pixels of the image. For the resize attribute, you also can specify the height as a percentage (an integer followed by the percent (%) symbol). When you resize an image, if you specify a value for the width, you can let ColdFusion calculate the aspect ratio by specifying &quot;&quot; as the height. If specified, the value must be an integer.</td>
<td></td>
</tr>
</tbody>
</table>
| interpolation | resize | Optional | highestQuality | Specified a specific interpolation algorithm by name (for example, hamming), by image quality (for example, mediumQuality), or by performance (for example, highestPerformance). The following are the valid values:

- highestQuality
- highQuality
- mediumQuality
- highestPerformance
- highPerformance
- mediumPerformance
- nearest
- bilinear
- bicubic
- bessel
- blackman
- hamming
- hanning
- hermite
- lanczos
- mitchell
- quadratic

| isBase64 | border convert info read resize rotate write writeToBrowser | Optional | no | Specifies whether the source is a Base64 string:

- yes: the source is a Base64 string.
- no: the source is not a Base64 string. |
<table>
<thead>
<tr>
<th>name</th>
<th>action</th>
<th>type</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>borderconvertreadresizerotate</td>
<td>Optional (see Description)</td>
<td>Name of the ColdFusion image variable to create. The read action requires a name attribute. The border, resize, and rotate actions require a name attribute or a destination attribute. You can specify both.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>overwrite</td>
<td>bordercaptchaconvertreadresizerotatewriter</td>
<td>Optional</td>
<td>no</td>
<td>Valid only if the destination attribute is specified. The overwrite values are: • yes: overwrites the destination file. • no: does not overwrite the destination file. If the destination file exists, ColdFusion generates an error if the overwrite action is not set to yes.</td>
</tr>
<tr>
<td>quality</td>
<td>write</td>
<td>Optional</td>
<td>0.75</td>
<td>Quality of the JPEG destination file. Applies only to files with an extension of JPG or JPEG. Valid values are fractions that range from 0 through 1 (the lower the number, the lower the quality).</td>
</tr>
<tr>
<td><strong>source</strong></td>
<td><strong>border convert info read resize rotate write writeToBrowser</strong></td>
<td><strong>Required</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------</td>
<td>--------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>structName</strong></td>
<td><strong>info</strong></td>
<td><strong>Required</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>text</strong></td>
<td><strong>captcha</strong></td>
<td><strong>Required</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **URL of the source image**: for example, "{{ http://www.google.com/images/logo.gif }}"
- **Absolute pathname or a pathname relative to the web root**: for example: "c:\images\logo.jpg"
- **ColdFusion image variable containing another image, BLOB, or byte array**: for example, "#{myImage}"
- **Base64 string**: for example, "data:image/jpg;base64,/9j/4AAQS...kZJRgABAQA......" 

- **Name of the ColdFusion structure to be created**.
- **Text string displayed in the CAPTCHA image**: Use capital letters for better readability. Do not include spaces because users cannot detect them in the resulting CAPTCHA image.
### Usage

ColdFusion provides the `cfimage` tag and the ColdFusion image, a construct native to ColdFusion that contains image data. You can manipulate ColdFusion images in memory and write them to a file, a database, or directly to a browser. You use the `cfimage` tag to create ColdFusion images from existing image files and perform simple image actions, such as rotating or resizing. Alternatively, you can use the `ImageNew` function to create a ColdFusion image from the beginning or from an existing image. You can use the `Image` functions to perform complex image manipulation operations on ColdFusion images that you create with the `cfimage` tag or with the `ImageNew` function.

You can perform the following tasks with ColdFusion images:

- Convert an image from one file format to another. For example, you can convert a BMP file to a JPEG file or a Base64 string to a GIF.
- Enforce consistent sizes on files uploaded to the server.
- Enforce size limits on JPEG images (by changing the quality of the image).
- Save a ColdFusion image to a file or write the image directly to a browser.
- Use the `ImageGetBlob` function within the `cfquery` tag to insert a ColdFusion image as a Binary Large Object Bitmap (BLOB) in a database. Also, you can extract a BLOB from a database and generate a ColdFusion image from it.
- Create watermark images.
- Create thumbnail images.
- Create a Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) image.

<table>
<thead>
<tr>
<th>attribute</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>thickness</td>
<td>Border thickness in pixels. The border is added to the outside edge of the source image, increasing the image area accordingly. The value must be an integer.</td>
</tr>
<tr>
<td>border</td>
<td>Optional</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>attribute</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>Width in pixels of the image. For resize, you also can specify the width as a percentage (an integer followed by the % symbol). When you resize an image, if you specify a value for the height, you can let ColdFusion calculate the aspect ratio by specifying &quot;&quot; as the width. If specified, the value must be an integer.</td>
</tr>
<tr>
<td>captcha resize</td>
<td>Required</td>
</tr>
</tbody>
</table>
a distorted text image that is human-readable, but not machine-readable, used in a challenge-response test for preventing spam.
For more detailed examples, see Creating and Manipulating ColdFusion Images in the Developing ColdFusion Applications.

File attributes

Use the following syntax to specify an in-memory file, which is not written to disk in the destination and source attributes. In-memory files speed processing of transient data.

```
ram:///filepath
```

The filepath can include directories, for example `ram:///petStore/images/poodle.jpg`. Create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications.

Supported image file formats

The `cfimage` tag operates on a number of different file formats. To list the formats that are supported on the server where the ColdFusion application is deployed, use the `GetReadableImageFormats` function and the `GetWritableImageFormats` function. ColdFusion supports the following default image formats on Macintosh, Windows, and Unix operating systems:

- JPEG
- GIF
- TIFF
- PNG
- BMP

ColdFusion does not support the following image formats:

- Animated GIF
- Multipage TIFF
- PSD
- AI

CMYK support

The `cfimage` tag supports reading and writing CMYK images, but does not support actions that require converting the images. For example, you can use CMYK images with the `read`, `write`, `writeToBrowser`, `resize`, `rotate`, and `info` actions. You cannot use CMYK images with the `convert`, `captcha`, and `border` actions. The same rule applies to image functions. For example, the `ImageNew`, `ImageRead`, and `ImageWrite` functions support CMYK images, but the `ImageAddBorder` function does not.

Valid HTML named colors

The following table lists the W3C HTML 4 named color value or hexadecimal values that the `color` attribute accepts:

<table>
<thead>
<tr>
<th>Color name</th>
<th>RGB value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>##000000</td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Color</th>
<th>Hexadecimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>##0000FF</td>
</tr>
<tr>
<td>Red</td>
<td>##FF0000</td>
</tr>
<tr>
<td>Gray</td>
<td>##808080</td>
</tr>
<tr>
<td>LightGray</td>
<td>##D3D3D3</td>
</tr>
<tr>
<td>DarkGray</td>
<td>##A9A9A9</td>
</tr>
<tr>
<td>Green</td>
<td>##008000</td>
</tr>
<tr>
<td>Pink</td>
<td>##FFC0CB</td>
</tr>
<tr>
<td>Cyan</td>
<td>##00FFFF</td>
</tr>
<tr>
<td>Magenta</td>
<td>##FF00FF</td>
</tr>
<tr>
<td>Orange</td>
<td>##FFA500</td>
</tr>
<tr>
<td>White</td>
<td>##FFFFFF</td>
</tr>
<tr>
<td>Yellow</td>
<td>##FFFF00</td>
</tr>
</tbody>
</table>

For all other color values, enter the hexadecimal value. Enter a six-digit value, which specifies the RGB value. Values between 00 and FF are allowed.

**Image quality**

By default, the `cfimage` tag generates images with antialiasing turned on (to remove the appearance of jagged edges). The interpolation method is set to `highestQuality`: this produces a high-quality image, but decreases processing speed. To turn off antialiasing, use the `ImageSetAntialiasing` function. To change the interpolation method or for more control over image attributes, use the following functions:

- `ImageResize`
- `ImageRotate`
- `ImageScaleToFit`
- `ImageShear`
- `ImageTranslate`

**border action** Use the `border` action to create a rectangular border around the outer edge of an image. You can control the thickness of the border and its color. For more control, use the `ImageAddBorder` function. The following example shows how to set the thickness and color of a border:
• **captcha action** Use the `captcha` action to create a distorted text image that is human-readable but not machine-readable. When you create a CAPTCHA image, you specify the text that is displayed in the CAPTCHA image; ColdFusion randomly distorts the text. You can specify the height and width of the text area, which affects the spacing between letters, the font size, the fonts used for the CAPTCHA text, and the level of difficulty, which affects readability. The following example shows how to write a CAPTCHA image directly to the browser:

```coldfusion
<cfimage action="captcha" fontSize="25" width="400" height="150" text="rEadMe" fonts="Arial,Verdana,Courier New">  
```

**Note**

For the CAPTCHA image to display, the `width` value must be greater than: `fontSize` times the number of characters specified in `text` times 1.08. In this example, the minimum width is 162.

ColdFusion supports CAPTCHA images in PNG format only.

**Note**

Use unique names for the CAPTCHA image files so that when multiple users access the CAPTCHA images, the files are not overwritten.

The following example shows how to create CAPTCHA images with a medium level of difficulty that are written to files:

```coldfusion
<!--- Use the GetTickCount function to generate unique names for the CAPTCHA files. --->
<cfset tc = GetTickCount()>
<cfimage tc = GetTickCount()>
<cfimage action="captcha" fontSize="15" width="180" height="50" text="rEadMe" destination="images/rEadMe#tc#.png" difficulty="medium">  
```

For a detailed example, see the [Creating and Manipulating ColdFusion Images](#) in the *Developing ColdFusion Applications*.

• **convert action** Use the `convert` action to convert an image from one file format to another. For more information on file formats, see the section Supported image file formats. The following example shows how
to convert a JPEG file to a PNG file:

```html
<!--- This example shows how to convert a JPEG image to a PNG image. --->
<cfimage source="../cfdocs/images/artgallery/aiden02.jpg" action="convert" destination="aiden02.png">

**Note**

Converting images between one file format to another is time-consuming. Also, image quality can degrade; for example, PNG images support 24-bit color, but GIF images support only 256 colors. Converting transparent images (images with alpha) can degrade image quality.

- **info action** Use the `info` action to create a ColdFusion structure that contains information about the image, including the color model, height, width, and source of the image. The structure is the same as returned by the `ImageInfo` function. The following example shows how to retrieve all of the information about an image:

```html
<!--- This example shows how to retrieve and display image information. --->
<cfimage source="../cfdocs/images/artgallery/viata03.jpg" action="info" structName="viatoInfo">
<cfdump var="#viatoInfo#">

<!--- Alternatively, you can use the `cfoutput` tag to display specific image information, as shown in the following example. --->
<cfoutput>
<p>height: #viatoInfo.height# pixels</p>
<p>width: #viatoInfo.width# pixels</p>
<p>source: #viatoInfo.source#</p>
<p>transparency: #viatoInfo.colormodel.transparency#</p>
<p>pixel size: #viatoInfo.colormodel.pixel_size#</p>
<p>color model: #viatoInfo.colormodel.colormodel_type#</p>
<p>alpha channel support: #viatoInfo.colormodel.alpha_channel_support#</p>
<p>color space: #viatoInfo.colormodel.colorspace#</p>
</cfoutput>
```

- **read action** Use the `read` action to read an image from the specified local file pathname or URL, and create a ColdFusion image in memory. You can use the ColdFusion image variable as the source for another `cfimage` tag or for `Image` functions. The `read` action performs the same operation as the `ImageRead` function. The following example shows how to create a ColdFusion image from a JPEG file and manipulate it using the `ImageGrayscale` function:
• **resize action** Use the `resize` action to resize an image to the specified height and width. You can specify the height and width in pixels or as a percentage:

```cfml
<!--- This example shows how to specify the height and width of an image in pixels. --->
cfimage source="../cfdocs/images/artgallery/jeff01.jpg" action="resize" width="100" height="100" destination="jeff01_thumbnail.jpg" overwrite="yes">
<!--- This example shows how to specify the height and width of an image as percentages. --->
cfimage source="../cfdocs/images/artgallery/jeff02.jpg" action="resize" width="50%" height="50%" destination="jeff02_thumbnail.jpg" overwrite="yes">
<!--- This example shows how to specify the height of an image in pixels and its width as a percentage. --->
cfimage source="../cfdocs/images/artgallery/jeff03.jpg" action="resize" width="50%" height="100" destination="jeff03_thumbnail.jpg" overwrite="yes">
```

For more control of resize attributes, use the `ImageResize` function.

• **rotate action** Use the `rotate` action to rotate an image by degrees:

```cfml
<!--- This example shows how to rotate an image by 30 degrees. --->
cfimage source="../cfdocs/images/artgallery/maxwell01.jpg" action="rotate" angle="30" name="maxwellAngle">
<!--- Display the rotated image in a browser. --->
cfimage source="#maxwellAngle#" action="writeToBrowser">
```

For more control of the rotate attributes, use the `ImageRotate` function.

• **write action** Use the `write` action to write an image to the specified path. The new image is converted to the file type specified in the destination attribute. The `write` action performs the same operation as the `ImageWrite` function. When you write an image to a JPEG file, the image quality is set to 75% of the original image by default. To control the image size, use the `quality` attribute of the `write` action.

You can use the `write` action to change the quality of a JPEG image to reduce file size. The following example shows how to change image quality to .5:

```cfml
<cfimage source="#myImage#" action="write" destination="myGrayscaleImage.jpg" overwrite="yes">
```
writeToBrowser action Use the `writeToBrowser` action to display one or more ColdFusion images directly to the browser without writing them to files. Images are displayed in PNG format. The following example shows how to reduce the size of an image and display it in the browser:

Example

This example shows how to create a ColdFusion image and manipulate it by using `Image` functions:

```coldfusion
<!--- Create the ColdFusion image variable "myImage" from a JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Pass the ColdFusion image to the Image functions to blur the image by a radius of 5,
    flip the image 90 degrees, and convert the image to grayscale. --->
<cfset ImageBlur(myImage,5)>
<cfset ImageFlip(myImage,"90")>
<cfset ImageGrayscale(myImage)>
<!--- Write the transformed image to a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```
**cfimap**

**Description**

Queries an IMAP server to retrieve and manage mails within multiple folders.

**Category**

*Communications tags, Internet protocol tags*

**Syntax**

```
<cfimap
    action ="DELETE|DELETEFOLDER|CREATEFOLDER|OPEN|CLOSE|RENAMEFOLDER|LISTFOLDERS|MARKREAD|MOVEMAIL|GETALL|GETHEADERONLY"
    attachmentpath = "string"
    connection = "string"
    folder = "string"
    generateuniquefilenames = "yes|no"
    maxrows = "integer"
    messagenumber ="integer"
    name = "string"
    newfolder = "string"
    password = "string"
    port = "integer"
    recurse = "true|false"
    secure = "yes|no"
    server = "IMAP server address"
    startrow = "integer"
    secure = "yes|no"
    stoponerror = "true|false"
    uid = "integer or comma-delimited list of integers"
    username = "SMTP user ID">
```

**See Also**

*cfmailparam, cfmailparam, cfmailpart, cfpop, cfhttp, cfldap, Wrap: Using ColdFusion with mail servers* in Sending and Receiving E-Mail in the Developing ColdFusion Applications

**History**

New tag introduced in ColdFusion 9.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Req/Opt</td>
<td></td>
<td>A list of actions to perform on the mail.</td>
</tr>
<tr>
<td>attachmentpath</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>connection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>folder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>generateuniquefilenames</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxrows</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>messagenumber</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>newfolder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>password</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>port</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recurse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>secure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>startrow</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stoponerror</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>username</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>action</td>
<td>Optional</td>
<td>GetHeaderOnly</td>
<td>Returns the message header information for all retrieved mail. Other values for this attribute are: GetAll: Returns mail. The information includes the message header information, message text, and any attachments. Set the AttachmentPath attribute to retrieve attachments. Delete: Deletes messages from a folder. Open: Initiates an open session or connection with the IMAP server. Close: Terminates the open session or connection with the IMAP server. MarkRead: Marks all messages read from a folder. CreateFolder: Creates a folder in the Mailbox. DeleteFolder: Deletes a folder in the Mailbox. RenameFolder: Renames an existing user-defined folder. ListAllFolders: Displays a list of all existing folders in the mailbox or under the folder name defined by the Folder attribute. MoveMail: Moves mail from one folder to another.</td>
</tr>
<tr>
<td>attribute</td>
<td>value</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>attachmentpath</td>
<td>Optional for GetAll action</td>
<td>Specifies the name of the folder where ColdFusion retrieves attachments. If this folder does not exist, ColdFusion creates. If you do not specify the folder, then the attachment is not saved. Note: If you specify a relative path, then attachments are saved in a temporary folder. This folder is same as one returned by the GetTempDirectory function.</td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td>Required for the following actions: Open and Close</td>
<td>Specifies the variable name for the connection/session. If the server attribute has an invalid IP address or invalid domain name, then the connection fails and ColdFusion returns an error message.</td>
<td></td>
</tr>
<tr>
<td>Folder</td>
<td>Required for the following actions: DeleteFolder, CreateFolder, and RenameFolder Optional for ListAllFolders, Movemail, Markread, GetAll, GetHeaderonly, Delete.</td>
<td>For mail actions: Specifies the folder name where messages are retrieved, moved, or deleted. If folder name is invalid, ColdFusion defaults to INBOX. For folder actions: Specifies the folder name that is deleted (DeleteFolder) or created (CreateFolder) or renamed (RenameFolder). When selecting a subfolder, use the period (.) character as appropriate. For example, when deleting mail in folder Module in folder Product.Version.Module, define the Folder attribute as Product.Version.Module.</td>
<td></td>
</tr>
<tr>
<td>GenerateUniqueFileNames</td>
<td>Optional</td>
<td>Ensures that unique file names are generated for each attachment file. The goal is to avoid name conflicts for attachments that have the same filename.</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>MaxRows</td>
<td>Optional</td>
<td>Specifies the number of rows to be marked as read, deleted, or moved across folders. When the value is 1, it signals the row determined by StartRow. Any incremental value marks rows starting from the StartRow. If you have specified the UID or MessageNumber attribute, then MaxRows is ignored.</td>
<td></td>
</tr>
<tr>
<td>MessageNumber</td>
<td>Optional</td>
<td>Specifies the message number or a comma delimited list of message numbers for retrieval, deletion, marking mail as read, or moving mails. If you set an invalid message number, then the IMAP operation is ignored. For example, if you specify that cfimap deletes a specified message number, and if that message number does not exist, then the operation is ignored. If you have specified the UID attribute, then MessageNumber attribute is ignored.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Optional</td>
<td>Required for the following actions: GetAll, GetHeaderOnly, and ListAllFolders. Specifies the name for the query object that contains the retrieved message information.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NewFolder</td>
<td>Optional/Required for the following actions: RenameFolder, MoveMail. Specifies the name of the new folder when you rename a folder or the name of the destination folder where all mails move.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>Optional/Required when action=&quot;open&quot; username and password must be specified. Specifies the password for assessing the users' e-mail account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td>Optional 143 or 993. Specifies the IMAP port number. Use 143 for non-secure connections and 993 for secured connections.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recurse</td>
<td>Optional False. Specifies whether ColdFusion runs the CFIMAP command in subfolders. Recurse works for action=&quot;ListAllFolders&quot;. When recurse is set to &quot;true&quot;, ColdFusion parses through all folders and subfolders and returns folder/subfolder names and mail information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure</td>
<td>Optional False. Specifies whether the IMAP server uses a Secure Sockets Layer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td>Optional/Required for the Open action. Specifies the IMAP server identifier. You can assign a host name or an IP address as the IMAP server identifier.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>StartRow</td>
<td>Optional. Defines the first row number for reading or deleting. If you have specified the UID or MessageNumber attribute, then StartRow is ignored. You can also specify StartRow for moving mails.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
StopOnError | Optional | True | Specifies whether to ignore the exceptions for this operation. When the value is true, it stops processing, displays an appropriate error.

Timeout | Optional | 60 | Specifies the number of seconds to wait before timing out connection to IMAP server. An error message is displayed when timeout occurs.

Uid | Optional | Specifies the unique ID or a comma-delimited list of Uids to retrieve, delete, and move mails. If you set invalid Uids, then they are ignored.

Username | Optional | Specifies the user name. Typically, the user name is same the e-mail login.

Usage

Open a session or connection with an IMAP server. To open a session, define the server, user name, and password attributes. You can open a connection with an IMAP server by specifying values for the server, user name, password, and connection attributes. For a secure connection, specify secure="true". You can reuse the connection attribute in subsequent CFIMAP tags, without having to specify the server, user name, or password attributes. Once you have established a connection, you can perform the following actions:

- Retrieve mail: Retrieve mail using the GetHeaderOnly or GetAll attributes and store the information in a query object. Use the cfdump command to display the content of the query object. You can also download attachments in temporary ColdFusion folder or a new folder as defined by the AttachmentPath attribute.
- Delete any unnecessary mail or delete folders. You can delete any user-created folders. Standard folders, such as INBOX, OUTBOX, SEND, cannot be deleted.
- Mark multiple mail as read.
- Manage mail folders by creating folders, renaming them or moving mail across folders. If you are using sub folders, then use periods (.) to specify the exact path.

Once you have performed all actions, close the session or connection with the IMAP server. For example, mail from your e-mail account in Gmail can be retrieved by setting a connection to the Gmail IMAP server. You can define the login (user name) and set a secure connection. Next, you can quickly retrieve a top-level snapshot of e-mails using the GetHeaderOnly attribute or access full information about e-mails using the GetAll attribute.

Note

Gmail is not a complete IMAP implementation so some of the features of a regular IMAP server may not work with Gmail.
The following table lists the query information (column names) returned by various cfimap attributes.

<table>
<thead>
<tr>
<th>Values for &quot;action&quot; attribute</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHeaderOnly</td>
<td>ANSWERED, CC, DELETED, DRAFT, FLAGGED, FROM, HEADER, LINES, MESSAGEID, MESSAGENUMBER, RECENT, REPLYTO, RXDDATE, SEEN, SENTDATE, SIZE, SUBJECT, TO, UID</td>
</tr>
<tr>
<td>GetAll</td>
<td>ANSWERED, ATTACHMENTFILES, ATTACHMENTS, BODY, CC, CIDS, DELETED, DRAFT, FLAGGED, FROM, HEADER, HTMLBODY, LINES, MESSAGEID, MESSAGENUMBER, RECENT, REPLYTO, RXDDATE, SEEN, SENTDATE, SIZE, SUBJECT, TEXTBODY, TO, UID. CID is used to find the correct place of an image in an HTML e-mail message that the CFIMAP tag retrieves. If the e-mail message contains more than one embedded image, only the last embedded image is available.</td>
</tr>
<tr>
<td>ListAllFolders</td>
<td>FULLNAME (specifies the entire directory structure), NAME, NEW, TOTALMESSAGES, and UNREAD</td>
</tr>
</tbody>
</table>

⚠️ **Note**

The cfimap command works best on IMAP4 revision1. IMAP4 revision1 is backwards compatible with IMAP2 and IMAP2bis versions. Any previous versions are no more actively used.

You can get errors in following scenarios:

- Accessing an invalid server connection is established. Check the network conditions and whether you are using appropriate server IP address and domain names. Use valid e-mail user names and passwords.
- Accessing non-existent folders: Check whether the folder you are accessing exists. Create or rename folders with valid names. You cannot rename core folders. Move mail within existing folders.
- Slow network: Verify if the timeout attribute needs a higher value. Actions such as CreateFolder may need longer time to execute. In such cases, adjust the value of the timeout attribute.
- Incorrect usage of cfimap attributes: Check if you are using the correct attribute. For example, if you have 15 e-mails in a folder and if the startrow or maxrow attribute has value of 18 then ColdFusion returns an error.
- The e-mail client does not recognize IMAP access. Verify whether your e-mail is set up to allow IMAP access. Complete the necessary IMAP access in connection settings section of your e-mail client.
- Using incorrect syntax for attributes: Verify that all attributes are defined per syntax.

**Example:** 1
<!--- Retrieving e-mails from a folder --->
<html>
<head>
<title>IMAP Mail Client</title>
</head>
<body>
<!--- Replace your username and password with valid IMAP email account name and password. Replace "server address" with your IMAP server address--->
<cfimap
   server = "server address" <!------>
   username = "yourname"
   action="open"
   secure="yes"
   password = "yourpassword"
   connection = "test.cf.gmail">
<!--- Retrieve header information from the mailbox. --->
<cfimap
   action="getHeaderOnly"
   connection="test.cf.gmail"
   name="queryname">
<cfdump var="#queryname#">
<cfimap
   action="close"
   connection = "test.cf.gmail">
</body>
</html>

Example: 2
<!---- Create a folder; copy mail from Inbox and list all folder info---->
<html>
<head>
<title>IMAP Mail Client</title>
</head>
<body>
<!--- Replace yourname and yourpassword with valid IMAP email account name and password. Replace "server address" with your IMAP server address---->
<cfimap
    server = "server address"
    username = "yourname"
    action="open"
    secure="yes"
    password = "yourpassword"
    connection = "test.cf.gmail">
<!--- Create a new folder, named Folder1 --->
<cfimap
    action="CreateFolder"
    folder="Folder1"
    connection="test.cf.gmail">
<!--- Move first 2 mails from INBOX to Folder1. --->
<cfimap
    action="MoveMail"
    newfolder="Folder1"
    messagenumber="1,2"
    stoponerror="true"
    connection="test.cf.gmail">
<!--- List all folders and get the information in a query. --->
<cfimap action="listallfolders"
    connection="test.cf.gmail"
    name="queryname">
<!--- Display query containing all folders information. --->
<cfdump var="#queryname#">
<cfimap
    action="close"
    connection = "test.cf.gmail">
</body>
</html>

Example: 3
<!--- Use form-based entry to access cfimap mail account. Save the form with name login.cfm. --->
<html>
<head>
<title>IMAP Mail Client</title>
</head>
<body>
<cfif IsDefined("form.server")>
<!--- Make sure server, username are not empty. --->
<cfif form.server is not "" and form.username is not "">
<cfimap
    server = "#form.server#"
    username = "#form.username#"
    action="open"
    Secure="yes"
    password = "#form.pwd#"
    connection = "#form.server#">
<cfimap
    action="ListAllFolders"
    connection="#form.server#"
    name="queryname">
<h3>Folders in your Inbox <cfoutput>#form.username#</cfoutput></h3>
<ol>
<cfoutput query = "queryname">
<li>#NAME# - #TOTALMESSAGES# <b>(#UNREAD#)</b></li>
</cfoutput>
</ol>
</cfimap
</cfif>
</cfelse>
<form action = "login.cfm " method = "post">
<table>
<tr>
<td>Enter IMAP mail server</td><td><input type = "Text" name = "server"></td>
</tr>
<tr>
<td>Enter your username</td><td><input type = "Text" name = "username"></td>
</tr>
<tr>
<td>Enter your password</td><td><input type = "password" name = "pwd"></td>
</tr>
<tr>
<td colspan="2"><input type="Submit" value="Get Folder List" name="getFolderList"></td>
</tr>
</table>
</form>
</cfif>
</body>
</html>
**cfimapfilter**

**Description**

Specifies filter parameters that control the actions of `cfimap.get` operations.

**History**

ColdFusion 11: Added this tag.

**Category**

Communications tags

**Syntax**

```
<cfimapfilter
   name = "filter type"
   value = "filter value">
```

OR

```
<cfimapfilter
   name = "filter type"
   from = "date/time"
   to = "date/time">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfimap, Getting Exchange items and attachments in the Developing ColdFusion Applications

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>The type of filter to use. The supported filters in ColdFusion 11 are:</td>
</tr>
<tr>
<td>Subject</td>
<td>Value</td>
<td>String to search in Subject</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td>From</td>
<td>Value</td>
<td>From ID</td>
<td></td>
</tr>
<tr>
<td>To</td>
<td>Value</td>
<td>Recipient ID</td>
<td></td>
</tr>
<tr>
<td>Flag</td>
<td>Value</td>
<td>Flag Description Value</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>Unseen Messages that do not have the Seen flag set.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time Received</th>
<th>From, To</th>
<th>Start and End date for Time Received</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The start date or date/time combination of the range to use for filtering. Cannot be used with the <code>value</code> attribute. If you specify a <code>from</code> attribute without a <code>to</code> attribute, the filter selects for all entries on or after the specified date or time. The value can be in any date/time format recognized by ColdFusion, but must correspond to a value that is appropriate for the filter type.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>To</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The end date or date/time combination for the range used for filtering. Cannot be used with the <code>value</code> attribute. If you specify a <code>to</code> attribute without a <code>from</code> attribute, the filter selects for all entries on or before the specified date or time. The value can be in any date/time format recognized by ColdFusion, but must correspond to a value that is appropriate for the filter type.</td>
</tr>
</tbody>
</table>
| value | Optional | The filter value for all filters that do not take a date or time range. Cannot be used with the `from` and `to` attributes. ColdFusion generates an error if you specify this attribute with an empty contents. Therefore, you cannot use the empty string to search for empty values. If you set this attribute to the empty string (""), ColdFusion searches for entries where the specified field is empty.

---

**Example**

```cfc
<cfimap action="getall"
   connection = "#REQUEST.connectionname#"
   name = "queryname"
   stoponerror = "#REQUEST.stoponerror#" >

   <cfimapfilter name="subject" value="filter"/>

</cfimap>
```
**cfimpersonate**

**Description**

This tag is obsolete. Use the newer security tools; see [Conversion functions](#) and [Securing Applications](#) in the *Developing ColdFusion Applications*.

**History**

ColdFusion MX: This tag is obsolete. It does not work in ColdFusion MX and later releases.
**cfimport**

**Description**

You can use the `cfimport` tag to import either of the following:

- All ColdFusion pages in a directory, as a tag custom tag library.
- A Java Server Page (JSP) tag library. A JSP tag library is a packaged set of tag handlers that conform to the JSP 1.1 tag extension API.

**Category**

*Application framework tags*

**Syntax**

```cfml
<cfimport
    path = "the CFC namespace to import"
    prefix = "custom"
    taglib = "tag library location">
```

**See also**

*cfapplication*

**History**

ColdFusion 9: Added the path attribute.

ColdFusion MX: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Required</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>taglib</td>
<td>Required</td>
<td>Tag library URI. The path must be relative to the web root (and start with /), the current page location, or a directory specified in the Administrator ColdFusion mappings page. A directory in which custom ColdFusion tags are stored. In this case, all the cfm pages in this directory are treated as custom tags in a tag library. A path to a JAR in a web-application, for example, &quot;/WEB-INF/lib/sometags.jar&quot; A path to a tag library descriptor, for example, &quot;/sometags.tld&quot; <strong>Note</strong>: You must put JSP custom tag libraries in the /WEB-INF/lib directory. This limitation does not apply to ColdFusion pages.</td>
<td></td>
</tr>
<tr>
<td>prefix</td>
<td>Required</td>
<td>Prefix by which to access the imported custom CFML tags JSP tags. If you import a CFML custom tag directory and specify an empty value, &quot;,&quot;, for this attribute, you can call the custom tags without using a prefix. Specify and use a prefix for a JSP tag library.</td>
<td></td>
</tr>
<tr>
<td>path</td>
<td>Optional</td>
<td>Use this attribute to import a CFC namespace.</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

The following example imports the tags from the directory myCustomTags:
You can import multiple tag libraries using one prefix. If there are duplicate tags in a library, the first one takes precedence. JSP tags have fixed attributes; however, if the tag supports runtime attribute expressions, most tag libraries support the use of the syntax \#expressions\#. To reference a JSP tag in a CFML page, use the syntax `<prefix:tagname>`.

Set the prefix value in the prefix attribute.

### Use JSP custom tags in a ColdFusion page

1. Put a JSP tag library JAR file (for example, myjsptags.jar) into the ColdFusion server directory `wwwroot/WEB-INF/lib`. If the tag library has a separate TLD file, put it in the same directory as the JAR file.
2. At the top of a CFML page, insert code such as the following:

```cfml
<cfimport
    prefix="mytags"
    taglib="/WEB-INF/lib/myjsptags.jar">
```

To reference a JSP tag from a JAR file, use the following syntax:

```cfml
<cfoutput>
    <mytags:helloTag message="#mymessage#" />
<cfoutput>
```

The `cfimport` tag must be on the page that uses the imported tags. For example, if you use a `cfimport` tag on a page that you include with the `cfinclude` call, you cannot use the imported tags on the page that has the `cfinclude` tag. Similarly, if you have a `cfimport` tag on your `Application.cfm` page, the imported tags are available on the `Application.cfm` page only, not on the other pages in the application. ColdFusion does not throw an error in these situations, but the imported tags do not run. You cannot use the `cfimport` tag to suppress output from a tag library.

For more information, see the Java Server Page 1.1 specification.
<h3>cfimport example</h3>
<p>This example uses the random JSP tag library that is available from the Jakarta Taglibs project, at http://jakarta.apache.org/taglibs/</p>

<cfimport taglib="/WEB-INF/lib/taglibs-random.jar" prefix="randomnum">

<randomnum:number id="randPass" range="000000-999999" algorithm="SHA1PRNG" provider="SUN" />
<cfset myPassword = randPass.random>
<cfoutput>
Your password is #myPassword#<br>
</cfoutput>

You can also instantiate a component using the new operator:

<cfset a = new abc()>
**cfinclude**

**Description**

Embeds references to ColdFusion pages in CFML. You can embed cfinclude tags recursively. For another way to encapsulate CFML, see cfmessagebox. (A ColdFusion page was formerly sometimes called a ColdFusion template or a template.)

**Category**

Flow-control tags, Page processing tags

**Syntax**

```cfml
<cfinclude
    template = "template name"
    runOnce = "true|false">
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfcache,cfflush,cfheader,cfhtmlhead,cfsetting,cfsilent

**History**

ColdFusion 11: The compileextforinclude attribute of the `<cfapplication>` tag affects the behavior of the `<cfinclude>` tag.

- By default only files with the cfm and cfml extensions get compiled when included using the `<cfinclude>` tag. All other files when included using the cfinclude tag will not get compiled but their content will be statically included. No error will be thrown.
- A server-level setting and an application-level setting has been added to specify a list of file extensions that can contain the CFM code. While processing the `<cfinclude>` tag, ColdFusion checks for this Application-level/server-level setting to decide whether to compile the content or to statically include it. The setting value is comma separated list of file extensions. If file extensions are specified at application-level, the server-level file extension list will be ignored and the application-level file extension list will be used.
- Specifying a special value wildcard (*) in the file extension list makes the cfinclude tag to compile any file.
- The server-level setting can be modified using the ColdFusion Administrator. The setting can be found at Server Settings > Settings page. A new key/tag attribute compileextforinclude added to the application cfc/cfm to specify this file extension list.

ColdFusion 10: Added the attribute {{runOnce}}

ColdFusion MX: Changed error behavior: if you use this tag to include a CFML page whose length is zero bytes, you do not get an error.

**Attributes**
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template</td>
<td>Required</td>
<td></td>
<td>A logical path to a ColdFusion page.</td>
</tr>
<tr>
<td>runOnce</td>
<td>Optional</td>
<td>false</td>
<td>If set to true, the given template (if already processed) is not processed again for a given request.</td>
</tr>
</tbody>
</table>

Usage

ColdFusion searches for included files in the following locations:

1. In the directory of the current page or a directory relative to the current page
2. In directories mapped in the ColdFusion Administrator

You cannot specify an absolute URL or file system path for the file to include. You can only use paths relative to the directory of the including page or a directory that is registered in the ColdFusion Administrator Mappings. The following cfinclude statements work, assuming that the myinclude.cfm file exists in the specified directory:

```cfc
<cfinclude template="myinclude.cfm"
<cfinclude template="../myinclude.cfm"
<cfinclude template="/CFIDE/debug/myinclude.cfm"
```

But the following do not work:

```cfc
<cfinclude template="C:\ColdFusion\wwwroot\doccomments\myinclude.cfm"
<cfinclude template="http://localhost:8500/doccomments/myinclude.cfm"
```

The included file must be a syntactically correct and complete CFML page. For example, to output data from within the included page, you must have a cfoutput tag, including the end tag, on the included page, not the referring page. Similarly, you cannot span a cfif tag across the referring page and the included page; it must be complete within the included page.

You can specify a variable for the template attribute, as the following example shows:

```cfc
<cfset templatetouse="../header/header.cfm"
<cfinclude template="#templatetouse#"
```

Example
<!--- This example shows the use of cfinclude to paste CFML or HTML code into another page dynamically. --->

<h4>This example includes the dochome.htm page from the CFDOCS directory. The images do not display, because they are located in a separate directory. However, the page appears fully rendered within the contents of this page.</h4>
<cfinclude template = "../cfdocs/dochome.htm"/>
**cfindex**

**Description**

Populates a search engine collection with metadata and creates indexes for searching it. The engine (Solr) can search physical files of various types or a database query. Indexing database columns that result from a query lets users search the query data much faster than they could if you used multiple SQL queries to return the same data. You must define a collection using the ColdFusion Administrator or the `cfcollection` tag before creating indexes for the collection.

You also can index a collection using the ColdFusion Administrator. For more information on creating, indexing, and searching a collection, see Building a Search Interface in the *Developing ColdFusion Applications*.

**Category**

**Extensibility tags**

**Syntax**

```
cfindex supports script style syntax:

new index(collection="<collection_name>");

<cfindex
  action = "update|delete|purge|refresh|fullimport|deltainport|status|abort"
  autoCommit = "yes|no"
  collection = "collection name"
  body = "body"
  category = "category name"
  categoryTree = "category tree"
  custom1 = "custom value"
  custom2 = "custom value"
  custom3 = "custom value"
  custom4 = "custom value"
  extensions = "file extensions"
  key = "ID"
  language = "{{language}}"
  prefix = "location of documents"
  query = "query name"
  recurse = "yes|no"
  status = "status"
  title = "title"
  type = "type"
  URLpath = "URL">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

`cfcollection`, `cfexecute`, `cfojbject`, `cfreport`, `cfsearch`, `cfwddx`
History

ColdFusion 10:
- Added new actions `fullimport`, `deltaimport`, `status`, `abort`
- Added the attribute `autoCommit`, `fieldBoost`, `docBoost`

ColdFusion 9: Added Solr search engine support.

ColdFusion MX 7.0.1: Added the `prefix` attribute.

ColdFusion MX 7:
- Added the `category`, `categoryTree`, `custom3`, and `custom4` attributes for the `update` and `refresh` actions.
- Added the `status` attribute for the `update`, `refresh`, `delete`, and `purge` actions.
- Removed reference to external collections.
- Removed suggested `cflock` usage.
  - ColdFusion MX:
    - The `action` attribute value `optimize` is obsolete. It does not work, and might cause an error, in ColdFusion MX.
    - Changed the `external` attribute behavior: it is not necessary to specify the `external` attribute. (ColdFusion automatically detects whether a collection is internal or external.)
    - Changed Verity operations behavior: ColdFusion supports Verity operations on Acrobat PDF files.
    - Changed thrown exceptions: this tag can throw the `SEARCHENGINE` exception.
    - Changed acceptable collection naming: this tag accepts collection names that include spaces.
    - Changed query result behavior: the `cfindex` tag can index the query results from a `cfsearch` tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

| action                  | Required                                      | update: updates a collection and adds key to the index.  
delete: removes collection documents as specified by the key attribute.  
purge: deletes all of the documents in a collection. Causes the collection to be taken offline, preventing searches.  
refresh: deletes all of the documents in a collection, and then performs an update.  
fullimport: To index full database. For instance, when you index the database for the first time.  
deltaimport: For partial indexing. For instance, for any updates in the database, instead of a full import, you can perform delta import to update your collection.  
status: Provides the status of indexing, such as the total number of documents processed and status such as idle or running.  
abort: aborts an ongoing indexing task.  |
|------------------------|-----------------------------------------------|-------------------------------------------------------------------------------------------------|
| autoCommit             | Optional                                      | If yes, automatically commit the changes to the search server.  
If no, indexed documents are not committed unless you specifically commit using cfindex action="commit". By default, the value is set to true. |
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collection</td>
<td>Required</td>
<td>Name of a collection that is registered by ColdFusion; for example, &quot;personnel&quot;.</td>
</tr>
<tr>
<td>body</td>
<td>Required if <code>type=custom</code></td>
<td>ASCII text to index. Query column names, if name is specified in query. You can specify columns in a delimited list, for example: &quot;emp_name, dept_name, location&quot;. This attribute is ignored if <code>type</code> is <code>file</code> or <code>path</code>, and is invalid if <code>action</code> is <code>delete</code>.</td>
</tr>
<tr>
<td>category</td>
<td>Optional</td>
<td>A string value that specifies one or more search categories for which to index the data. You can define multiple categories, separated by commas, for a single index. If you use a query for your indexed data, this value can be the name of a column.</td>
</tr>
<tr>
<td>categoryTree</td>
<td>Optional</td>
<td>A string value that specifies a hierarchical category or category tree for searching. It is a series of categories separated by forward slashes (&quot;/&quot;). You can specify only one category tree.</td>
</tr>
<tr>
<td>Option</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>custom1</td>
<td>Optional</td>
<td>Use to index discrete values in collection records, which lets you search for specific records. By contrast, values specified in the body attribute are concatenated and searched as a body of text using the specified criteria. If type = custom, a query column name. If type is file or path, a string.</td>
</tr>
<tr>
<td>custom2</td>
<td>Optional</td>
<td>Usage is the same as for custom1.</td>
</tr>
<tr>
<td>custom3</td>
<td>Optional</td>
<td>Usage is the same as for custom1.</td>
</tr>
<tr>
<td>custom4</td>
<td>Optional</td>
<td>Usage is the same as for custom1.</td>
</tr>
<tr>
<td>docBoost</td>
<td>Optional</td>
<td>Boost entire document while indexing. docBoost enhances the score of the documents and thereby the ranking in the search results.</td>
</tr>
<tr>
<td>extensions</td>
<td>Optional</td>
<td>Delimited list of file extensions that ColdFusion uses to index files, if type = &quot;Path&quot;. &quot;.&quot; returns files with no extension. &quot;.&quot; returns all files. For example, the following code returns files with a listed extension or no extension: extensions = &quot;.htm, .html, .cfm, .cfml, *.&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extensions</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>htm, html, cfm, cfm, dbm, dbml</td>
<td>Delimited list of file extensions that ColdFusion uses to index files, if type = &quot;Path&quot;. &quot;.&quot; returns files with no extension. &quot;.&quot; returns all files. For example, the following code returns files with a listed extension or no extension: extensions = &quot;.htm, .html, .cfm, .cfml, *.&quot;</td>
</tr>
<tr>
<td>fieldBoost</td>
<td>Optional</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| key             | Required | (empty string)                | The value specified for key depends on the type attribute:  
- If type = "file", the directory path and filename for the file,  
- If type = "path", the directory path for the location of the files.  
- If type = "custom", a unique identifier that specifies the location of the data. For a query, the name of the column that holds the primary key, for example. If not a query, an identifier such as the URL for a web page, for example. |
| language        | Optional | English                       | For options, see cfcollection. |
| prefix          | Optional |                               | Specifies the location of files to index when the computer that contains the K2 Search Service is not the computer on which you installed ColdFusion, and when you index files with the type attribute set to path. |
| query           | Optional |                               | The name of the query against which the collection is generated. |
### cfindex Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| `recurse` | Optional | `no`    | - `yes` if `type = "path"`, indexes qualified files in directories below the path specified in the `key` attribute.  
- `no` |
| `status`  | Optional |         | The name of the structure into which ColdFusion returns status information. |
| `title`   | Optional |         | Provides a title for the document if one cannot be extracted from the document. |
| `type`    | Optional | `custom`, if `query attribute` is specified. Otherwise, `file`. | - `file`: applies action value to filename, including path. Expects a filename in the `key` attribute.  
- `path`: applies action to files in a directory path that pass the extensions filter. Expects a directory name in the `key` attribute.  
- `dih`: used for Data Import Handler.  
- `custom`: applies action to custom data; for example, to data from a query. |
| `URLpath` | Optional |         | If `type` is `file` or `path`, specifies the URL path. During indexing, this pathname is prefixed to filenames and returned from a search as the `url`. |

### Usage

The attributes settings that the `cfindex` tag requires depend on whether you set the `query` attribute. If you set the query attribute to a valid query name, it specifies that `cfindex` is to index the data in the query rather than indexing documents on a disk. If you do not set the query attribute, `cfindex` assumes that it is indexing a file (`type = file`), a set of files in a directory path (`type = path`), or text that you provide in the `body` attribute (`type = custom`). If you set the query attribute to a valid query name, the `cfindex` tag creates indexes as specified by the following attributes and their values:
<table>
<thead>
<tr>
<th>Type</th>
<th>Attribute values</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>The <code>key</code> attribute is the name of a column in the query that contains a full filename (including path).</td>
</tr>
<tr>
<td>Path</td>
<td>The <code>key</code> attribute is the name of a column in the query that contains a directory pathname.</td>
</tr>
<tr>
<td></td>
<td>The <code>extensions</code> and <code>recurse</code> attributes, if specified, elaborate on which files are included. If the action is delete, <code>cfindex</code> deletes keys for the collection.</td>
</tr>
<tr>
<td>Custom</td>
<td>The <code>key</code> attribute specifies a column name that contains anything you want; for example, the primary key value in the database. It must be unique because this is the primary key in the collection. If the action is delete, the <code>key</code> attribute is the name of a column in the query that contains the keys to delete.</td>
</tr>
<tr>
<td></td>
<td>The <code>body</code> attribute is required and is a comma-delimited list of the names of the columns that contain the text data to be indexed.</td>
</tr>
</tbody>
</table>

If you do not set the `query` attribute, the `cfindex` tag creates indexes as specified by the following attributes and their values:

<table>
<thead>
<tr>
<th>Type</th>
<th>Attribute values</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>The <code>key</code> attribute is required and is a full pathname to a file.</td>
</tr>
<tr>
<td>Path</td>
<td>The <code>key</code> attribute is required and it is a directory pathname.</td>
</tr>
<tr>
<td></td>
<td>The <code>extensions</code> and <code>recurse</code> attributes, if specified, designate which types of files are included. If the action is delete, both the keys and the document files are deleted.</td>
</tr>
<tr>
<td>Custom</td>
<td>The <code>key</code> attribute is an identifier that specifies the key. If the action is delete, the <code>key</code> attribute is the document key to delete.</td>
</tr>
<tr>
<td></td>
<td>The <code>body</code> attribute is required and is the text to be indexed.</td>
</tr>
</tbody>
</table>

If `type` is not specified but `query` is set, ColdFusion sets the `type` to the default value of `custom`. If neither `type` nor `query` is set, ColdFusion sets `type` to the default value of `file`. If `type` equals `custom`, all attributes except for
key and body can specify a literal value, not only a column name. This allows you to change a field to empty in the collection.

Status attribute

The status attribute provides the following information and diagnostics about the result of a cfindex operation:

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BADKEYS</td>
<td>Struct</td>
<td>A structure of keys with diagnostic messages about the indexing of these keys. If there are no bad keys, this key does not exist.</td>
</tr>
<tr>
<td>DELETED</td>
<td>Number</td>
<td>The number of keys deleted.</td>
</tr>
<tr>
<td>MESSAGES</td>
<td>Array</td>
<td>An array of diagnostic messages, including nonfatal errors and warnings. If there are no messages, this key does not exist.</td>
</tr>
<tr>
<td>INSERTED</td>
<td>Number</td>
<td>The number of keys inserted into the collection.</td>
</tr>
<tr>
<td>UPDATED</td>
<td>Number</td>
<td>The number of keys updated in the collection.</td>
</tr>
</tbody>
</table>

Example

```cfml
<!--- EXAMPLE #1 Index a file, type = "file". ---------------------------->
<!--- Example dumps content of status variable (info). ------------------->
<cfindex collection="CodeColl" action="refresh" type="file"
  key="C:\ColdFusion\wwwroot\vw_files\cfindex.htm"
  urlpath="http://localhost:8500/vw_files/"
  language="English"
  title="Cfindex Reference page"
  status="info">
  <!--- Search for Attributes. --->
  <cfsearch
    name = "mySearch"
    collection = "CodeColl"
    criteria = "Attributes"
    contextpassages = "1"
    maxrows = "100">
    <cfoutput>
      key=#mySearch.key#<br />
      title=#mySearch.title#<br />
      context=#mySearch.context#<br />
      url=#mySearch.url#<br />
    </cfoutput>
</cfsearch>
```
<!-- EXAMPLE #2 Index a path (type = "path"). -->
<cfindex collection="CodeColl"
action="refresh"
type="path"
key="C:\inetpub\wwwroot\vw_files\newspaper\sports"
urlpath="http://localhost/vw_files/newspaper/sports"
extensions = ".htm, .html"
recurse="no"
language="English"
categoryTree="vw_files/newspaper/sports"
category="Giants">

<!-- Search for any references to criteria. -->
<cfsearch
name = "mySearch"
collection = "CodeColl"
categoryTree="vw_files/newspaper/sports"
category="Giants"
criteria = "Williams"
contextpassages = "1"
maxrows = "100">
<cfoutput>
key=#mySearch.key#\n<br />
title=#mySearch.title#\n<br />
context=#mySearch.context#\n<br />
url=#mySearch.url#\n<br />
</cfoutput>

<!-- EXAMPLE #3: Index a QUERY (type = "custom") using custom1. -->
<!-- Retrieve data from the table. -->
<cfquery name="getCourses" datasource="cfdocexamples">
SELECT * FROM COURSES
</cfquery>

<!-- Update the collection with the above query results. -->
<!-- key is Course_ID in the Courses table. -->
<!-- body specifies the columns to be indexed for searching. -->
<!-- custom1 specifies the value of the Course_Number column. -->
<cfindex
query="getCourses"
collection="CodeColl"
action="Update"
type="Custom"
key="Course_ID"
title="Courses"
body="Course_ID,Descript"
custom1="Course_Number">
<h2>Indexing Complete</h2>
<!-- cno supplies value for searching custom1; could be form input instead. -->
<cfset cno = "540">
<cfsearch
collection = "CodeColl"
criteria = "CF_CUSTOM1 <MATCHES> #cno#"
contextpassages = "1"
maxrows = "100">
<cfoutput>
key=#mySearch.key#<br />
title=#mySearch.title#<br />
context=#mySearch.context#<br />
url=#mySearch.url#<br />
</cfoutput>
</cfquery>

<!---- EXAMPLE #4 Index a FILE within a QUERY (type= "file"). ------------------>
<!---- Retrieve row with a column that contains a filename (Contract_File). ---->
<cfquery name="getEmps" datasource="cfdocexamples">
SELECT * FROM EMPLOYEE WHERE EMP_ID = 1
</cfquery>

<!---- Update the collection with the above query results. ---->
<!---- key specifies the column that contains a complete filename. ---->
<!---- file is indexed in same way as if no query involved. ---->
<cfindex
query="getEmps"
collection="CodeColl"
action="Update"
type="file"
key="Contract_File"
title="Contract_File"
body="Emp_ID,FirstName,LastName,Contract_File">

<h2>Indexing Complete</h2>
<cfsearch
name = "mySearch"
collection = "CodeColl"
criteria = "vacation"
contextpassages = "1"
maxrows = "100">
<cfoutput>
key=#mySearch.key#<br />
title=#mySearch.title#<br />
context=#mySearch.context#<br />
url=#mySearch.url#<br />
</cfoutput>
</cfquery>

<!---- EXAMPLE # 5 Index a PATH within a QUERY. ----------------------------
<!---- Retrieve a row with a column that contains a path (Project_Docs). ---->
<cfquery name="getEmps" datasource="cfdocexamples">
SELECT * FROM EMPLOYEE WHERE Emp_ID = 15
</cfquery>

<!---- Update the collection with the above query results. ---->
<!---- key specifies a column that contains a directory path. ---->
<!---- path is indexed in same way as if no query involved. ---->
<cfindex
query="getEmps"
collection="CodeColl"
action="update"
type="path"
key="Project_Docs"
title="Project_Docs"
<h2>Indexing Complete</h2>

<cfsearch>
  name = "getEmps"
  collection = "CodeColl"
  criteria = "cfsetting"
  contextpassages = "1"
  maxrows = "100"
</cfsearch>

<cfoutput>
  key=#getEmps.key#<br />
  title=#getEmps.title#<br />
  context=#getEmps.context#<br />
  url=#getEmps.url#<br />
</cfoutput>

<!--- EXAMPLE #6 Deletes keys in the CodeColl collection for html files --->
<!--- in the specified directory (but not in subdirectories). --------------->

<cfindex collection="CodeColl"
  action="delete"
  type="path"
  key="C:\ColdFusion\wwwroot\vw_files\newspaper"
  urlpath="http://localhost:8500/vw_files/newspaper"
  extensions = ".htm, .html"
  recurse="no">

<!--- EXAMPLE #7 Purges all keys in the CodeColl collection --->
<!--- with recursion. ----------------------------------------------->

<cfindex collection="CodeColl"
  action="purge"
type="path"
key="C:\ColdFusion\wwwroot\vw_files\newspaper"
cfinput

Description

Used within the cfform tag, to place input controls that support input validation on a form.

Category

Forms tags

Syntax
Some attributes apply to only specific display formats. For details, see the Attributes table. In HTML format forms, standard HTML input control attributes not listed above are passed to the HTML and have their normal effect.
Note

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also
cfajaximport, cfapplet, cfcalendar, cfform, cfformgroup, cfformitem, cfgrid, cfselect, cfslider, cftextarea, cftree, Using Ajax form controls and features in Using Ajax User Interface Components and Features in Developing ColdFusion Applications

History

ColdFusion 8

- Added autosuggest, autosuggestBindDelay, autosuggestMinLength, delimiter, maxResultsDisplayed, showAutosuggestLoadingIcon, and typeahead attributes.
- Added support for the bind attribute in HTML forms and the bindAttribute and bindOnload, and bindOnSubmit attributes.
- Added the sourceForTooltip attribute
- Added support for datefield value of the type attribute in HTML forms.

ColdFusion MX 7:

- Added support for button, file, hidden, image, reset, and submit controls.
- Added support for generating Flash and XML controls (specified in the cfform tag).
- Added datefield type (Flash forms only) and the supporting dayNames and monthNames attributes.
- Added bind, enabled, height, label, tooltip, visible, and width attributes for use in Flash forms.
- Added support for onBlur and onServer validation, including the validateAt attribute.
- Added USdate, range, boolean, email, URL, uuid, guid, maxlength, noblanks, and submitOnce validate attribute values.
- Added support for preventing multiple submissions.
- Added the mask attribute.
- Deprecated the passthrough attribute. The tag now supports all HTML input tag attributes directly.

ColdFusion MX: Changed the cfform tag preserveData attribute behavior: if it is set to True, ColdFusion checks radio and check box values only if their value matches the posted value for the control. (In earlier releases, if the posted value did not match any of the cfinput check boxes or radio buttons for the control, the checked attribute was used.

Attributes

The following table lists attributes that ColdFusion uses directly. The tag also supports all HTML form tag attributes that are not on this list, and passes them directly to the browser.

Note

Attributes that are not marked as supported in All or XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt; formats</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required; all</td>
<td></td>
<td>Name for form input element.</td>
</tr>
</tbody>
</table>
| autosuggest    | Optional, HTML   |         | Specifies entry completion suggestions to display as the user types into a text input. The user can select a suggestion to complete the text entry. The valid value can be either of the following:  
  • A string that consists of the suggestion values separated by the delimiter specified by the `delimiter` attribute.  
  • A bind expression that gets the suggestion values based on the current input text. The bind expression must pass a `cfautosuggest` bind parameter to represent the user input.  
  Valid only for `cfinput type="text"`. For more information, see Using autosuggest text input fields in the *Developing ColdFusion Applications* |
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autosuggestBindDelay</td>
<td>Optional, HTML</td>
<td>1 second</td>
<td>A nonzero integer that specifies the minimum time between autosuggest bind expression invocations, in seconds. This value also specifies the delay from when the user first enters a minimum-length entry in the field until the suggestion box appears. Use this attribute to limit the number of requests that are sent to the server when a user types. Valid only for <code>cfinput</code> type=&quot;text&quot;. <strong>Note:</strong> The only way to get the default behavior is to omit the attribute. Otherwise, the delay must be a nonzero integer value.</td>
</tr>
<tr>
<td>autosuggestMinLength</td>
<td>Optional, HTML</td>
<td>1</td>
<td>The minimum number of characters required in the text box before invoking a bind expression to return items for suggestion. Valid only for <code>cfinput</code> type=&quot;text&quot;.</td>
</tr>
<tr>
<td>bind</td>
<td>Optional; HTML, Flash</td>
<td></td>
<td>A bind expression that dynamically sets an attribute of the control. For details, see Usage.</td>
</tr>
<tr>
<td>bindAttribute</td>
<td>Optional; HTML</td>
<td>value</td>
<td>Specifies the HTML tag attribute whose value is set by the bind attribute. You can only specify attributes in the browser's HTML DOM tree, not ColdFusion-specific attributes. Ignored if there is no bind attribute. Valid only for <code>cfinput</code> type=&quot;text&quot;.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>bindOnLoad</td>
<td>Optional; HTML</td>
<td>no</td>
<td>A Boolean value that specifies whether to execute the <code>bind</code> attribute expression when first loading the form. Ignored if there is no <code>bind</code> attribute. Valid only for <code>cfinput type=&quot;text&quot;</code>.</td>
</tr>
</tbody>
</table>
| checked         | Optional; all | no    | Selects a radio button or check box control:  
|                 |            |       | • yes  
|                 |            |       | • no  
<p>|                 |            |       | For HTML format, you can indicate that the item is selected by specifying the value as <code>checked</code> or specifying the attribute with no value. |
| dayNames        | Optional; all | S, M, T, W, T, F, S | Applies to <code>datefield</code> type only. A comma-delimited list that sets the names of the weekdays displayed in the calendar. Sunday is the first day; the rest of the weekday names follow in the normal order. |
| delimiter       | Optional, HTML | comma (,) | The delimiter to use to separate entries in a static auto-suggest list. This attribute is meaningful only if the <code>autosuggest</code> attribute is a string of delimited values. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| disabled        | Optional; all Not disabled Disables user input, making the control read-only. The attribute behavior depends on the format of the form as follows:  
- HTML format forms: ColdFusion passes this attribute directly to the HTML. To disable input, specify disabled without a value (HTML format) or with the value disabled (XHTML compliant). To enable input, omit this attribute.  
- Flash format forms: To disable input, specify disabled without an attribute, or disabled = "yes" (or any ColdFusion positive Boolean value, such as true). To enable input, omit the attribute or specify disabled="no" (or any ColdFusion negative Boolean value, such as false). |
<p>| enabled         | Optional; Flash Yes Boolean value that specifies whether the control is enabled. A disabled control appears in light gray. The inverse of the disabled attribute. |
| firstDayOfWeek  | Optional; all 0 Integer in the range 0-6 that specifies the first day of the week in the calendar: 0 indicates Sunday; 6 indicates Saturday. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>Optional; see Description</td>
<td>Applies to most Flash types, HTML image type on some browsers. The height of the control, in pixels. The displayed height might be less than the specified size.</td>
</tr>
<tr>
<td>id</td>
<td>Optional; HTML name attribute value</td>
<td>The HTML ID of the form.</td>
</tr>
<tr>
<td>label</td>
<td>Optional; Flash, XML</td>
<td>Label to put next to the control on a Flash form. Not used for button, hidden, image, reset, or submit types.</td>
</tr>
</tbody>
</table>
| mask      | Optional; Flash, HTML | For tags with type="text". A mask pattern that controls the character pattern that users can enter, or that the form sends to ColdFusion. Mask characters and the corresponding valid input characters are:  
  - A = A-Za-z  
  - X = A-Za-z0-9  
  - 9 = 0-9  
  - ? = Any character  
  - All other characters = insert the literal character  
  For tags with type="textfield", a pattern that controls the format of dates that the user selects in the calendar. Mask characters are:  
  - D = day; can use 0-2 mask characters.  
  - M = month; can use 0-4 mask characters.  
  - Y = year; can use 0, 2, or 4 characters.  
  - E = day in week; can use 0-4 characters.  
  For more information, see the Usage section. |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchContains</td>
<td>Optional</td>
<td>false</td>
<td>If true, the match returned contains the query string. By default, the results that start with the query string are returned.</td>
</tr>
<tr>
<td>maxLength</td>
<td>Optional; all</td>
<td></td>
<td>Maximum length of text entered, if type = &quot;Text&quot; or &quot;password&quot;. For complete length validation, specify maxLength validation in a validate attribute; otherwise, this attribute prevents users from typing beyond the specified length, but does not prevent them from pasting in a longer value.</td>
</tr>
<tr>
<td>maxResultsDisplayed</td>
<td>Optional; HTML</td>
<td>10</td>
<td>The maximum number suggestions to display in the auto-suggest list. Valid only for cfinput type=&quot;text&quot;.</td>
</tr>
<tr>
<td>message</td>
<td>Optional; all</td>
<td></td>
<td>Message text to display if validation fails.</td>
</tr>
<tr>
<td>monthNames</td>
<td>Optional; all</td>
<td></td>
<td>Applies to datefield type only. A comma-delimited list of the month names that display at the top of the calendar.</td>
</tr>
<tr>
<td>Event</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>onBindError</td>
<td>Optional; HTML</td>
<td>The name of a JavaScript function to execute if evaluating a bind expression, including an auto-suggest bind expression, results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and have specified a global error handler (by using the ColdFusion.setGlobalErrorHandler function), it displays the error message; otherwise a default error pop-up displays.</td>
<td></td>
</tr>
<tr>
<td>onChange</td>
<td>Optional; all</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the control changes due to user action. In Flash, applies to datefield, password, and text types only.</td>
<td></td>
</tr>
<tr>
<td>onClick</td>
<td>Optional; all</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the user clicks the control. In Flash, applies to button, checkbox, image, radio, reset, and submit types only.</td>
<td></td>
</tr>
<tr>
<td>onError</td>
<td>Optional; HTML, XML</td>
<td>Name of a custom JavaScript function to execute if validation fails.</td>
<td></td>
</tr>
<tr>
<td>onKeyDown</td>
<td>Optional; all</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) ActionScript to run when the user presses a keyboard key in the control.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Required/Optional</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>onKeyUp</td>
<td>Optional; all</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a keyboard key in the control.</td>
<td></td>
</tr>
<tr>
<td>onMouseDown</td>
<td>Optional; all</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a mouse button in the control.</td>
<td></td>
</tr>
<tr>
<td>onMouseUp</td>
<td>Optional; all</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the user presses a mouse button in the control.</td>
<td></td>
</tr>
<tr>
<td>onValidate</td>
<td>Optional; HTML, XML</td>
<td>Name of a custom JavaScript function to validate user input. The form object, input object, and input object values are passed to the routine, which should return <code>true</code> if validation succeeds, and <code>false</code> otherwise. If used, the <code>validate</code> attribute is ignored.</td>
<td></td>
</tr>
<tr>
<td>pattern</td>
<td>Required if <code>validate= &quot;regex&quot;</code>; HTML, XML</td>
<td>JavaScript regular expression pattern to validate input. ColdFusion uses this attribute only if you specify <code>regex</code> in the <code>validate</code> attribute. Omit leading and trailing slashes. For examples and syntax, see Building Dynamic Forms with <code>&lt;cform&gt;</code> Tags in the Developing ColdFusion Applications.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Options</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>range</td>
<td>Optional; all</td>
<td></td>
<td>Minimum and maximum allowed numeric values. ColdFusion uses this attribute only if you specify range in the validate attribute. If you specify a single number or a single number followed by a comma, it is treated as a minimum, with no maximum. If you specify a comma followed by a number, the maximum is set to the specified number, with no minimum. <strong>Note:</strong> ColdFusion does not process the range attribute when you use onsubmit or onBlur validation in XML format forms.</td>
</tr>
<tr>
<td>readonly</td>
<td>Optional; Flash, HTML</td>
<td></td>
<td>Applies to HTML and Flash forms. Valid only for cfinput type= &quot;text&quot;. ColdFusion ignores this attribute for all other input types.</td>
</tr>
</tbody>
</table>
| required   | Optional; all             | no        | • yes: the field must contain data.  
• no: allows an empty field.                                                                                                                                                                                                                                                                                                             |
<p>| showAutosuggestLoadingIcon | Optional; HTML | true     | A Boolean value that specifies whether to display an animated icon when loading an auto-suggest value for a text input.                                                                                                                                                                                                                               |
| size       | Optional; all             |           | Size of input control. Ignored, if type = &quot;radio&quot; or &quot;checkbox&quot;. If specified in a Flash form, ColdFusion sets the control width pixel value to 10 times the specified size and ignores the width attribute.                                                                                                                                                                                                 |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sourceForTooltip</code></td>
<td>Optional; HTML</td>
<td>The URL of a page to display as a tool tip. The page can include HTML markup to control the format, and the tip can include images. If you specify this attribute, an animated icon appears with the text “Loading...” while the tip is being loaded.</td>
</tr>
<tr>
<td><code>src</code></td>
<td>Optional; Flash, HTML</td>
<td>Applies to Flash button, reset, submit, and image types, and the HTML image type. URL of an image to use on the button.</td>
</tr>
<tr>
<td><code>style</code></td>
<td>Optional; all</td>
<td>In HTML or XML format, ColdFusion passes the style attribute to the browser or XML. In Flash format, must be a style specification in CSS format. For detailed information on specifying Flash styles, see <a href="Creating%20Forms%20in%20Flash">Creating Forms in Flash</a> in the Developing ColdFusion Applications. In XML format, ColdFusion passes the style attribute to the XML.</td>
</tr>
<tr>
<td><code>tooltip</code></td>
<td>Optional; Flash, HTML</td>
<td>Text to display when the mouse pointer hovers over the control. Ignored if you specify a <code>sourceForTooltip</code> attribute.</td>
</tr>
</tbody>
</table>


| **type** | Optional;all | **text** | The input control type to create:  
- **button**: push button.  
- **checkbox**: check box.  
- **datefield**: HTML and Flash only; date entry field with an expanding calendar from which users select the date or dates. In HTML format only, users can also enter the date by typing in the field.  
- **file**: file selector; not supported in Flash. Not supported when you use Ajax form submission to submit the form asynchronously from the page.  
- **hidden**: invisible control.  
- **image**: clickable button with an image.  
- **password**: password entry control; hides input values.  
- **radio**: radio button.  
- **reset**: form reset button.  
- **submit**: form submission button.  
- **text**: text entry box. The attribute supports all HTML 5 input types, for example, email, range, or date. |
<p>| <strong>typeahead</strong> | Optional;HTML | <strong>no</strong> | A Boolean value that specifies whether the auto-suggest feature should automatically complete a user's entry with the first result in the suggestion list. Valid only for <code>cfinput type=&quot;text&quot;</code>. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>validate</td>
<td>Optional; all</td>
<td>The type or types of validation to do. Available validation types and algorithms depend on the format. For details, see Usage.</td>
</tr>
<tr>
<td>validateAt</td>
<td>Optional; all</td>
<td>How to do the validation; one or more of the following values: onSubmit, onServer, onBlur. The onBlur and onSubmit values are identical in Flash forms. For multiple values, use a comma-delimited list. For details, see Usage.</td>
</tr>
<tr>
<td>value</td>
<td>depends on type setting; all</td>
<td>HTML: corresponds to the HTML value attribute. Its use depends on control type. Flash: optional; specifies text for button type inputs: button, submit, and image.</td>
</tr>
<tr>
<td>visible</td>
<td>Optional; Flash</td>
<td>Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank.</td>
</tr>
<tr>
<td>width</td>
<td>Optional; see Description</td>
<td>Applies to most Flash types, and HTML image type on some browsers. The width of the control, in pixels. For Flash forms, ColdFusion ignores this attribute if you also specify a size attribute value.</td>
</tr>
</tbody>
</table>

⚠️ **Note**

Attributes that are marked as not supported in XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.
Usage

For this tag to work properly, the browser must be JavaScript-enabled. If the cfform preserveData attribute is true and the form posts back to the same page, the posted value of the cfinput control is used, instead of its Value or Checked attribute. You can use the keyboard to access and select dates from a datefield Flash input: press Tab to go to the field and press the Spacebar to open the menu. Use the Up, Down, Left, and Right Arrow keys to change the selected date. Use the Home and End keys to reach the first and last enabled date in a month, respectively. Use the Page Up and Page Down keys to reach the previous and next month, respectively.

Note

To clear a datefield entry in Flash format forms, select the field to open the menu, and click the selected date.

For more information, see cfform. For information on using JavaScript regular expressions with this tag, see Building Dynamic Forms with cfform Tags in the Developing ColdFusion Applications.

Validation

The following sections describe how to do validation in cfinput tags.

- **Validation methods** ColdFusion provides four methods of validation of cfinput text and password fields. You can specify one or a combination of the following in the validateAt attribute:
  - onSubmit: The form page on the browser includes JavaScript functions that perform validation before the form is submitted to the server. In Flash format forms, this option is identical to onBlur.
  - onBlur: In HTML format the form page on the browser includes JavaScript functions that perform validation when the field loses the focus. In Flash format, the attribute is equivalent to onSubmit. OnBlur validation uses the same algorithms as onSubmit validation. OnBlur validation was added in ColdFusion MX 7.
  - onServer: ColdFusion performs the validation on the server. Some onServer algorithms vary from the onSubmit algorithms. OnServer Date and Time validation allow more patterns than onSubmit validation. OnServer validation was added in ColdFusion MX 7, and automatically generates hidden fields to support the validation. You can also omit a validate attribute and specify the type of validation for the field in a separate hidden form field. This form of validation is equivalent to onServer validation, but it lets you specify separate messages for each validation that you do on the field. It is backward compatible with previous ColdFusion releases. For more information on hidden form field validation, see cfform and Validating form data using hidden fields in Validating form data using hidden fields in the Developing ColdFusion Applications.
  - Validation types Use the following values in the validate attribute to specify input validation for all validation methods. Most attributes apply only to password or text fields. You can specify multiple validation types in a comma-delimited list, but only some combinations are meaningful.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>If validateAt=&quot;onServer&quot;, allows any date format that returns true in the IsDate function; otherwise, same as USdate.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>USdate</td>
<td>A US date of the format mm/dd/yy mm-dd-yy or mm.dd.yy, with 1-2 digit days and months, 1-4 digit years.</td>
</tr>
<tr>
<td>eurodate</td>
<td>A date of the format dd/mm/yy, with 1-2 digit days and months, 1-4 digit years. The format can use /, -, or . characters as delimiters.</td>
</tr>
<tr>
<td>time</td>
<td>Time format hh:mm:ss</td>
</tr>
<tr>
<td>float or numeric</td>
<td>A number; allows integers.</td>
</tr>
<tr>
<td>integer</td>
<td>An integer.</td>
</tr>
<tr>
<td>range</td>
<td>A numeric range.</td>
</tr>
<tr>
<td>boolean</td>
<td>A value that can be converted to a Boolean value: Yes, No, True, False, or a number.</td>
</tr>
<tr>
<td>telephone</td>
<td>Standard U.S. telephone formats. Allows an initial 1 long-distance designator and up to 5-digit extensions, optionally starting with x.</td>
</tr>
<tr>
<td>zipcode</td>
<td>U.S. 5- or 9-digit ZIP code format #####-####. The separator can be a hyphen (-) or a space.</td>
</tr>
<tr>
<td>creditcard</td>
<td>Strips blanks and dashes; verifies number using mod10 algorithm. Number must have 13-16 digits.</td>
</tr>
<tr>
<td>ssn or social_security_number</td>
<td>US. Social Security number format, #######. The separator can be a hyphen (-) or a space.</td>
</tr>
<tr>
<td>email</td>
<td>A valid e-mail address of the form <a href="mailto:name@server.domain">name@server.domain</a>. ColdFusion validates the format only; it does not check that entry is a valid active e-mail address.</td>
</tr>
<tr>
<td>URL</td>
<td>A valid URL pattern; supports http, https, ftp file, mailto, and news URLs.</td>
</tr>
<tr>
<td>guid</td>
<td>A unique identifier that follows the Microsoft/DCE format, xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx, where x is a hexadecimal number.</td>
</tr>
<tr>
<td>uuid</td>
<td>A universally unique identifier (UUID) that follows the ColdFusion format, xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx, where x is a hexadecimal number.</td>
</tr>
</tbody>
</table>
### Validation differences

The preceding table describes the general validation behavior. The underlying validation code must differ depending on the validation method and the form type. As a result, the algorithms used vary in some instances, including the following:

- The validation algorithms used for date/time values vary between onSubmit/OnBlur and OnServer.
- The algorithms used for onSubmit/OnBlur validation in Flash vary from those used for HTML/XML format, and generally follow simpler rules.

The table describes the onSubmit/OnBlur behavior in HTML format. For detailed information on the OnServer validation algorithms, see Data validation types in About ColdFusion validation in the Developing ColdFusion Applications. For more information on validation, including discussions of the advantages and disadvantages of different validation types, see Validating data in the Developing ColdFusion Applications.

### Masking input data

In HTML and Flash forms, the mask attribute controls the format of data that can be entered into a text field or that is selected in a datefield input control calendar. In HTML format, it does not prevent users from typing a date that does not follow the mask into a datefield input control. You can combine masking and validation on a field.

In text fields, ColdFusion automatically inserts any literal mask characters, such as hyphen (-) characters in telephone numbers. Users type only the variable part of the field.

The following pattern enforces entry of a part number of the format EB-1234-c1-098765, where the user starts the entry by typing the first numeric character, such as 3. ColdFusion fills in the preceding EB prefix and all - characters. The user must enter four numbers, followed by two alphanumeric characters, followed by six numbers.

```
<cfinput type="text" name="newPart" mask="EB-9999-XX-999999"/>
```

**Note**

To force a pattern to be all-uppercase or all-lowercase, use the ColdFusion `UCase` or `LCase` functions in the action page.

For tags with `type="datefield"` (and `cfcalendar` tags), the number of pattern characters determines the format of the output when the user selects a date in the calendar, as follows:

<table>
<thead>
<tr>
<th>Mask</th>
<th>Pattern</th>
</tr>
</thead>
</table>
The following pattern specifies that the Flash forms sends the date selected by using a `datefield` input control to ColdFusion as text in the format 04/29/2004:

```
<cfinput name="stDate" type="datefield" label="date:" mask="mm/dd/yyyy"/>
```

**Flash form data binding**

The `bind` attribute lets you populate form fields by using the contents of other form fields. To specify text from another field in a Flash format `cfinput` tag `bind` attribute, use the following format: `{sourceTagName.text}`. For example, the following line uses the values from the `firstName` and `lastName` fields to construct an e-mail address. (The user can change or replace this value with a typed entry.)

```
<cfinput type="text" name="email" label="email"
 bind="{firstName.text}.{lastName.text}@mm.com"/>
```

**HTML form data binding**

The `bind` attribute lets you set `cfinput` attributes dynamically. For example, you can automatically fill an e-mail field text-input value based on name and domain field values. In HTML format, the `bind` attribute specifies a `bind expression`, which can have any for the following forms:

- A **Bind parameter** or string that contains one or more bind parameters. A bind parameter specifies a form control
value or other attribute and, optionally, an event. In its most basic form, a bind parameter consists of the `name` or `id` attribute of the control to which you are binding in braces ({ }). The value of the control attributes specified in the bind parameters determine the value of the `cfinput` control attribute.

- A CFC or JavaScript function, or URL, typically using one or more bind parameters as function parameters. The data returned by the function or URL sets the `cfinput` attribute value.

For details of using HTML form data binding, see Binding data to form fields in the Developing ColdFusion Applications.

Note

To bind to a `cfinput` control with `type` attribute of `button`, specify a bind event setting such as `click` in the bind expression of the control that binds to the button. The default event, `onChange`, has no effect.

Example: without binding

```cfc
cfform action = "cfinput.cfm"
  <cfinput type = "Text" name = "MyPhone"
    message = "Enter telephone number, formatted xxx-xxx-xxxx (e.g. 617-761-2000)"
    validate = "telephone" required = "yes">
    <font size = -1 color = red>Required</font>
  </cfinput>
  <cfinput type = "Text" name = "MyZip"
    message = "Enter zip code, formatted xxxxx or xxxxx-xxxx"
    validate = "zipcode" required = "yes">
    <font size = -1 color = red>Required</font>
  </cfinput>
  <cfinput type = "Text" name = "MyRange" range = "1,5"
    message = "You must enter an integer from 1 to 5"
    validate = "integer" required = "no">
  </cfinput>
  <cfinput type = "Text" name = "MyDate"
    message = "Enter a correctly formatted date (dd/mm/yy)"
    validate = "date" required = "no">
  </cfinput>
  <input type = "Submit" name = "" value = "send my information">
</cfform>
```

Example: with binding

```cfc
cfform action = "cfinput.cfm"
  <cfinput type = "Text" name = "MyPhone"
    message = "Enter telephone number, formatted xxx-xxx-xxxx (e.g. 617-761-2000)"
    validate = "telephone" required = "yes">
    <font size = -1 color = red>Required</font>
  </cfinput>
  <cfinput type = "Text" name = "MyZip"
    message = "Enter zip code, formatted xxxxx or xxxxx-xxxx"
    validate = "zipcode" required = "yes">
    <font size = -1 color = red>Required</font>
  </cfinput>
  <cfinput type = "Text" name = "MyRange" range = "1,5"
    message = "You must enter an integer from 1 to 5"
    validate = "integer" required = "no">
  </cfinput>
  <cfinput type = "Text" name = "MyDate"
    message = "Enter a correctly formatted date (dd/mm/yy)"
    validate = "date" required = "no">
  </cfinput>
  <input type = "Submit" name = "" value = "send my information">
</cfform>
```
The following example uses binding to generate a default e-mail address based on input controls with a first, last and domain names, and fills in the e-mail text-input field with the result.

The CFML page contains the following code:

```cfml
<h3>CFC bind using onkeyup() event</h3>
<ul>
  <li>When you type in the First Name field, the first initial appears in the Email control.</li>
  <li>Each key you enter in the Last Name and Domain fields appears in the Email control.</li>
</ul>

<cfform name="mycfform">
First Name<font color="#FF0000">*</font>:&nbsp;&nbsp;&nbsp;
<cfinput type="text" name="firstname" required="yes"><br>
Last Name<font color="#FF0000">*</font>:&nbsp;&nbsp;&nbsp;<br>
<cfinput type="text" name="lastname" required="yes"><br>
Domain<font color="#FF0000">*</font>:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<br>
<cfinput type="text" name="domain" required="yes"><br>
<br>
<cfinput type="text" name="email1"
bind="cfc:bindFcns.getEmailId({firstname@keyup},{lastname@keyup},
{domain@keyup})">
<br><br>
<font color="#FF0000">*</font> indicates the field is required.
</cfform>
```

The bindFcns CFC contains the following code:

```cfml
<cfcomponent>
<cffunction name="getEmailId" access="remote">
  <cfargument name="firstname" />
  <cfargument name="lastname" />
  <cfargument name="domain" />
  <cfreturn "#left(firstname,1)#.#lcase(arguments.lastname)#@#
  lcase(domain)#" />
</cffunction>
</cfcomponent>
```
**cfinsert**

**Description**

Inserts records in data sources from data in a ColdFusion form or form Scope.

**Category**

*Database manipulation tags*

**Syntax**

```
<cfinsert
  dataSource = "data source name"
  tableName = "table name"
  fetchClientInfo = "true|false"
  formFields = "formfield1, formfield2, ..."
  password = "password"
  tableOwner = "owner"
  tableQualifier = "table qualifier"
  username = "user name">
```

⚠️ **Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

`cfprocparam`, `cfprocresult`, `cfquery`, `cfqueryparam`, `cfstoredproc`, `cftransaction`, `cfupdate`

**History**

ColdFusion 10: Added the attribute `fetchClientInfo` and `clientInfo`. ColdFusion MX: Deprecated the `connec
tString`, `dbName`, `dbServer`, `dbtype`, `provider`, and `providerDSN` attributes. They do not work, and might cause an error, in releases later than ColdFusion 5.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataSource</td>
<td>Required</td>
<td></td>
<td>Data source; contains table.</td>
</tr>
<tr>
<td>clientInfo</td>
<td>Optional</td>
<td></td>
<td>Structure containing properties of the client to be set on the database connection.</td>
</tr>
<tr>
<td><strong>tableName</strong></td>
<td><strong>Required</strong></td>
<td><strong>Table in which to insert form fields.</strong> <strong>ORACLE drivers:</strong> must be uppercase. <strong>Sybase driver:</strong> case-sensitive. Must be the same case used when table was created</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>fetchClientInfo</strong></td>
<td><strong>Optional</strong></td>
<td><strong>false</strong></td>
<td><strong>If set true, returns a struct with the key-value pair passed by the last query.</strong></td>
</tr>
<tr>
<td><strong>formFields</strong></td>
<td><strong>Optional</strong></td>
<td><strong>(all on form, except keys)</strong></td>
<td><strong>Comma-delimited list of form fields to insert. If not specified, all fields in the form are included. If a form field is not matched by a column name in the database, ColdFusion throws an error. The database table key field must be present in the form. It may be hidden.</strong></td>
</tr>
<tr>
<td><strong>password</strong></td>
<td><strong>Optional</strong></td>
<td></td>
<td><strong>Overrides password specified in ODBC setup.</strong></td>
</tr>
<tr>
<td><strong>tableOwner</strong></td>
<td><strong>Optional</strong></td>
<td></td>
<td><strong>For data sources that support table ownership (such as SQL Server, Oracle, and Sybase SQL Anywhere), use this field to specify the owner of the table.</strong></td>
</tr>
<tr>
<td><strong>tableQualifier</strong></td>
<td><strong>Optional</strong></td>
<td></td>
<td><strong>For data sources that support table qualifiers, use this field to specify qualifier for table. The purpose of table qualifiers varies among drivers. For SQL Server and Oracle, qualifier refers to name of database that contains table. For Intersolv dBASE driver, qualifier refers to directory where DBF files are located.</strong></td>
</tr>
</tbody>
</table>
Example

```coldfusion
<!--- This example shows how to use cfinsert instead of cfquery to put data in a datasource. --->
<!--- If form.POSTED exists, we insert new record, so begin cfinsert tag. --->
<cfif IsDefined("form.posted")>
  <cfinsert dataSource = "cfdocexamples"
     tableName = "COMMENTS"
     formFields = "Email,FromUser,Subject,MessText,Posted">
  <h3><i>Your record was added to the database.</i></h3>
</cfif>

<!--- Use a query to show the existing state of the database. --->
<cfquery name = "GetComments" dataSource = "cfdocexamples">
  SELECT
    CommentID, EMail, FromUser, Subject, CommType, MessText, Posted, Processed
  FROM
    COMMENTS
</cfquery>

<html>
<head></head>
<h3>cfinsert Example</h3>
<p>First, show a list of the comments in the cfdocexamples datasource. --->
  <table>
    <tr>
      <td>From User</td><td>Subject</td><td>Comment Type</td><td>Message</td><td>Date Posted</td>
    </tr>
    <cfoutput query = "GetComments">
      <tr>
        <td valign = top><a href = "mailto:#Email#">#FromUser#</A></td>
        <td valign = top>#Subject#</td>
        <td valign = top>#CommType#</td>
        <td valign = top><font size = "-2">#Left(MessText, 125)#</font></td>
        <td valign = top>#Posted#</td>
      </tr>
    </cfoutput>
  </table>
<p>Next, well offer the opportunity to enter a comment: --->
  <form action = "cfinsert.cfm" method = "post"
```
name = "" value = "insert my comment">
</form>

⚠️ Note

The cfinsert tag internally uses parameterized queries.
**cfinterface**

**Description**

Defines an interface that consists of a set of signatures for functions. The interface does not include the full function definitions; instead, you implement the functions in a ColdFusion component (CFC). The interfaces that you define by using this tag can make up the structure of a reusable application framework.

**History**

ColdFusion 8: Added this tag.

**Category**

*Application framework tags, Extensibility tags*

**Syntax**

```xml
<cfinterface
    displayName = "descriptive name"
    extends = "interfaceName1[,interfaceName2]..."
    Hint = "hint text">
    <cffunction ...
    <cfargument ...
    <cfargument ...
    ...
    </cffunction>
    <cffunction ...
    ...
    </cffunction>
    ...
</cfinterface>
```

**See also**

*cfargument, cfcomponent, cffunction, GetComponentMetaData, IsInstanceOf*

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayName</td>
<td>Optional</td>
<td></td>
<td>A value to be displayed when using introspection to show a descriptive name for the interface.</td>
</tr>
</tbody>
</table>
extends Optional A comma-delimited list of one or more interfaces that this interface extends. Any CFC that implements an interface must also implement all the functions in the interfaces specified by this property. If an interface extends another interface, and the child interface specifies a function with the same name as one in the parent interface, both functions must have the same attributes; otherwise ColdFusion generates an error.

hint Optional Text to be displayed when using introspection to show information about the interface. The hint attribute value follows the syntax line in the function description.

Usage

The `cfinterface` tag declares a set of related functions that any ColdFusion component (CFC) that implements the interface must define. The interface specifies function signatures, but does not implement the functions; instead, the CFC that implements the interface must contain the full function definitions.

For example, you could create a create, read, update, and delete (CRUD) interface that defines the basic signatures of the four operations. All components that implement the interface must then conform to the interface signatures.

You can then implement the interface in different components to manage different types of data sources. Because all the components implement the same interface, you can ensure that you can easily replace one component with another, depending on the specific data source that an individual application requires.

You define an interface by creating a ColdFusion file with a .cfc extension and specifying the `cfinterface` tag as the first and only top-level tag in the file. The filename determines the interface name, so `myInterface.cfc` defines the `myInterface` interface. You can specify any attributes in the `cfinterface` tag; however, only the names listed in the Attributes table are meaningful to ColdFusion. The filename must not contain commas, or any periods except for the separator before the .cfc extension. Inside the `cfinterface` tag body, you specify the interface by declaring the functions of the interface. The interface definition must follow these basic rules:

- The `cfinterface` tag body can contain only `cffunction` tags and comments.
- The `cffunction` tag bodies can contain only `cfargument` tags, which declare the function arguments, and comments.
- The `cffunction` tag body is optional.

    The following example shows the general format of an interface definition:
<cfinterface extends="IBasicInterface">
  <cffunction name="hello" description="Should print a greeting containing the input argument or 'world'.">
    <cfargument name="whom" type="string" default="world"/>
  </cffunction>
  <cffunction name="calculateTwo" returnType="numeric" output="no" description="calculates a result using two numbers and returns the result">
    <cfargument name="first" type="numeric" required="yes"/>
    <cfargument name="second" type="numeric" required="no" default="0"/>
  </cffunction>
  <cffunction name="disclaimer"/>
</cfinterface>

This interface extends the IBasicInterface interface, so any component that implements this interface must also implement the methods of the IBasicInterface interface. This interface requires the component to implement the following three functions:

- A hello function that can optionally take a single string argument, which has a default value of "world".
- A calculateTwo function that takes one required numeric argument, has an optional numeric argument with a default value of 0, and must return a number.
- A disclaimer function that takes no arguments and returns any type.

The CFC that implements an interface specifies the interface name in the cfcomponent tag's implements attribute. It must implement all of the interface's methods as specified in the interface cffunction tags. The order of function arguments in the interface definition and the component definition must be identical.

The following table lists the attributes that you can use in the cffunction and cfargument tags, and describes the requirements and limitations on how you can use them in the interface definition and the component implementation:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Interface requirements</th>
<th>Implementation requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>cffunction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>access</td>
<td>Optional; only public is allowed</td>
<td>Optional; can be public or remote.</td>
</tr>
<tr>
<td>description</td>
<td>Optional</td>
<td>Can differ from value in interface.</td>
</tr>
<tr>
<td>displayName</td>
<td>Optional</td>
<td>Can differ from value in interface.</td>
</tr>
<tr>
<td>hint</td>
<td>Optional</td>
<td>Can differ from value in interface.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td>Must be identical to value in interface.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>output</td>
<td>Optional</td>
<td>Need not be identical to the value in the interface. Even if you omit the attribute in the interface, you can use it in the implementation. Similarly, you can use the attribute in the interface and omit it in the implementation.</td>
</tr>
<tr>
<td>returnType</td>
<td>Optional</td>
<td>Must be identical to value in interface; however, an omitted type option and an option value of any are equivalent and ColdFusion treats them as a match.</td>
</tr>
<tr>
<td>roles</td>
<td>Not allowed</td>
<td>Can be any valid value.</td>
</tr>
<tr>
<td>cfargument</td>
<td></td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>Optional</td>
<td>Must be identical to value in interface. If you omit this attribute in the interface, you can specify any value in the implementation.</td>
</tr>
<tr>
<td>displayName</td>
<td>Optional</td>
<td>Can differ from value in interface</td>
</tr>
<tr>
<td>hint</td>
<td>Optional</td>
<td>Can differ from value in interface</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td>Must be identical to value in interface.</td>
</tr>
<tr>
<td>required</td>
<td>Optional</td>
<td>If the interface specifies yes, this attribute must also specify yes. If the interface specifies no or omits this attribute, you can specify no or omit the attribute.</td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
<td>Must be identical to value in interface; however, an omitted type option and an option value of any are equivalent and ColdFusion treats them as a match.</td>
</tr>
</tbody>
</table>

A CFC can implement multiple interfaces.
Note

If a CFC implements multiple interfaces and two or more of the interfaces define functions with identical names, the signatures of these functions must be the same in all the interfaces; ColdFusion does not support function overloading.

ColdFusion uses the same rules to locate interfaces as it does to locate components. You can use the `GetCompontentMetaData` function to get information about an interface. Adobe recommends that you use a consistent technique for identifying interface names, for example, by always starting the file (and therefore interface) name with a capital I. Any component that implements only that single interface could have a similar name, for example the same root prefixed by a capital C. You could have an IresourceInfo.cfc interface file and a corresponding CresourceInfo.cfc component file, for example.

Example

The following example defines an IBasicMath interface with add, subtract, multiply, and divide operations. The integerMath CFC implements this interface by defining integer arithmetic versions of the operations. The testMath.cfm application uses the integerMath functions to do arithmetic calculations on two decimal numbers (using the values of pi and e). As an exercise, consider modifying the interface definition to take and return values of any type, and then implement a second CFC that uses the `PrecisionEvaluate` function to calculate arbitrary precision arithmetic and return the results. (These versions are omitted for brevity.) The IBasicMath.cfc file defines the interface as follows:

```coldfusion
<cfinterface>
<cffunction name = add returntype = "numeric" output = "no"
  description = "Add two values">
  <cfargument name = "first" type="numeric" required = "no" default = "0">
  <cfargument name = "second" type = "numeric" required = "no" default = "0">
</cffunction>
<cffunction name = subtract returntype = "numeric" output = "no"
  description = "Subtract two values">
  <cfargument name = "first" type = "numeric" required = "no" default = "0">
  <cfargument name = "second" type = "numeric" required = "no" default = "0">
</cffunction>
<cffunction name = multiply returntype = "numeric" output = "no"
  description = "Multiply two values">
  <cfargument name = "first" type = "numeric" required = "no" default = "0">
  <cfargument name = "second" type = "numeric" required = "no" default = "0">
</cffunction>
<cffunction name = divide returntype = "numeric" output = "no"
  description = "Divide two values">
  <cfargument name = "first" type = "numeric" required = "no" default = "0">
  <cfargument name = "second" type = "numeric" required = "no" default = "1">
</cffunction>
</cfinterface>
```

The integerMath.cfc file defines the integerMath component, which implements the IBasicMath interface, as follows:
<cfcomponent implements = "IBasicMath">
<cffunction name = add returntype = "numeric" output = "no" 
description = "Add two values">
<cfargument name = "first" type = "numeric" required = "no" default = "0">
<cfargument name = "second" type = "numeric" required = "no" default = "0">
<cfreturn Round(first + second)></cffunction>
<cffunction name = subtract returntype = "numeric" output = "no" 
description = "Subtract two values">
<cfargument name = "first" type = "numeric" required = "no" default = "0">
<cfargument name = "second" type = "numeric" required = "no" default = "0">
<cfreturn Round(first - second)></cffunction>
<cffunction name = multiply returntype = "numeric" output = "no" 
description = "Multiply two values">
<cfargument name = "first" type = "numeric" required = "no" default = "0">
<cfargument name = "second" type = "numeric" required = "no" default = "0">
<cfreturn Round(first * second)></cffunction>
<cffunction name = divide returntype = "numeric" output = "no" 
description = "Divide two values">
<cfargument name = "first" type = "numeric" required = "no" default = "0">
<cfargument name = "second" type = "numeric" required = "no" default = "1">
<cfreturn Round(first / second)></cffunction>
</cfcomponent>

The testMath.cfm file uses the integerMath component methods to calculate integer values, as follows:

<cfscript>
arguments = StructNew();
arguments.first = pi();
arguments.second = "2.718281828459045235360287471352";
</cfscript>
<cfobject name = "iMathObj" component = "integerMath">
<h3>Function Arguments</h3>
argument 1: #arguments.first#<br>
argument 2: #arguments.second#<br>
<h3>Addition</h3>
#iMathObj.add(argumentCollection = arguments)#
<h3>Subtraction</h3>
#iMathObj.subtract(argumentCollection = arguments)#
<h3>Multiplication</h3>
#iMathObj.multiply(argumentCollection = arguments)#
<h3>Division</h3>
#iMathObj.divide(argumentCollection = arguments)#
</cfoutput>
cfinvoke

Description

Does either of the following:

- Invokes a component method from within a ColdFusion page or component.
- Invokes a web service.
  - This tag works as follows:
  - Transiently instantiates a component or web service and invokes a method on it.
  - Invokes a method on an instantiated component or web service.
  - This tag can pass parameters to a method in the following ways:
     - With the cfinvokeargument tag
     - As named attribute-value pairs, one attribute per parameter
     - As a structure, in the argumentCollection attribute

Category

Extensibility tags

Syntax

<!---- Syntax 1: This syntax invokes a method of a component. Improve this example --->
<cfinvoke
  component = "component name or reference"
  method = "method name"
  returnVariable = "variable name"
  argumentCollection = "argument collection"
  ...>

OR

<!---- Syntax 2: This syntax can invoke a method of a component only from within the component. --->
<cfinvoke
  method = "method name"
  returnVariable = "variable name"
  argumentCollection = "argument collection"
  ...>

OR

<!---- Syntax 3: This syntax invokes a web service. --->
<cfinvoke
  webservice = "Web service name or WSDL URL"
  method = "operation name"
  password = "password"
  proxyPassword = "password for proxy server"
  proxyPort = "port on proxy server"
  proxyServer = "WSDL proxy server URL"  proxyUser = "user ID for proxy server"
  returnVariable = "variable name"
  refreshWSDL = "yes|no"
  servicePort = "WSDL port name"
  timeout = "request timeout in seconds"
  username = "user name"
  wsdl2javaArgs = "argument string">
OR

<!--- Syntax 4A: This syntax invokes a component. --->
This syntax shows instantiation with the cfobject tag.
This cfinvoke syntax applies to instantiating a component
with the cfobject tag and to instantiating a component
with the CreateObject function. --->
<cfobject
    component = "component name"
    name = "name for instantiated component">
    <cfinvoke
        <!--- Value is object name, within number signs. --->
        component =="#name of instantiated component#"
        method = "method name"
        returnVariable = "variable name"
        argumentCollection = "argument collection"
    ...
</cfobject>

OR

<!--- Syntax 4B: This syntax invokes a web service. --->
This syntax shows instantiation with the cfobject tag.
This cfinvoke syntax applies to instantiating a web service with the cfobject tag
and to
instantiating a web service with the CreateObject function. --->
<cfobject
    webservice = "web service name or WSDL URL"
    name = "name for instantiated object"
    (optional cfobject attributes)>
    <cfinvoke
        webservice =="#cfobject name attribute value#"
        method = "method name"
        password = "password"
        proxyPassword = "password for proxy server"
        proxyPort = "port on proxy server"
        proxyServer = "name or IP address of WSDL proxy server"
        proxyUser = "user ID for proxy server"
        returnVariable = "variable name"
        refreshWSDL = "yes|no"
        servicePort = "WSDL port name"
        timeout = "request time-out in seconds"
    ...
username = "user name">
wsdl2javaArgs = "argument string">

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfargument, cffunction, cfinvokeargument, cfolebject, cfproperty, cfreturn, CreateObject

History

ColdFusion 10: Added the wsVersion attribute.

ColdFusion 8: Added the following attributes: refreshWSDL, wsdl2javaArgs attributes.

ColdFusion MX 7: Added the servicePort attribute.

ColdFusion MX 6.1: Added the following attributes: timeout, proxyServer, proxyPort, proxyUser, and proxyPassword.

ColdFusion MX: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>argumentCollection</td>
<td>Optional</td>
<td></td>
<td>Name of a structure; associative array of arguments to pass to the method.</td>
</tr>
<tr>
<td>component</td>
<td>See Usage.</td>
<td></td>
<td>String or component object; a reference to a component, or component to instantiate.</td>
</tr>
<tr>
<td>input_params ...</td>
<td></td>
<td></td>
<td>Input parameters. For each named input parameter specify param Name=paramValue.</td>
</tr>
<tr>
<td>method</td>
<td>See Usage.</td>
<td></td>
<td>Name of a method. For a web service, the name of an operation.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Required</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td>Password set in the Administrator, if any</td>
<td></td>
</tr>
<tr>
<td>proxyPassword</td>
<td>Optional</td>
<td>http.proxyPassword system property, if any</td>
<td></td>
</tr>
<tr>
<td>proxyPort</td>
<td>Optional</td>
<td>http.proxyPort system property, if any</td>
<td></td>
</tr>
<tr>
<td>proxyServer</td>
<td>Optional</td>
<td>http.proxyHost system property, if any</td>
<td></td>
</tr>
<tr>
<td>proxyUser</td>
<td>Optional</td>
<td>http.proxyUser system property, if any</td>
<td></td>
</tr>
<tr>
<td>refreshWSDL</td>
<td>Optional</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>returnVariable</td>
<td>Optional</td>
<td>Name of a variable for the invocation result</td>
<td></td>
</tr>
<tr>
<td>servicePort</td>
<td>Optional</td>
<td>First port found in the WSDL</td>
<td></td>
</tr>
<tr>
<td>timeout</td>
<td>Optional</td>
<td>0 (no timeout)</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>username</td>
<td>Optional</td>
<td>User name set in the Administrator, if any</td>
<td>The user name to use to access the web service. If the <code>webservice</code> attribute specifies a web service name configured in the Administrator, overrides any user name specified in the Administrator entry.</td>
</tr>
<tr>
<td>webservice</td>
<td>See Usage</td>
<td></td>
<td>One of the following:</td>
</tr>
</tbody>
</table>
### wsVersion

**Optional**

Specifies version of the axis to be used. If the `wsVersion` is specified as "1" then axis 1 will be used for Consuming. If it is "2" then axis 2 will be used.

### Note

If you do not specify any attributes of the proxy server, and a corresponding system property is set (typically in the JVM startup arguments) ColdFusion uses the system property value.

### Usage

The following table shows when you can use each attribute:

<table>
<thead>
<tr>
<th>This attribute is required, optional, ignored, or invalid:</th>
<th>For this <code>cfinvoke</code> tag syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax 1</strong></td>
<td><strong>Syntax 2</strong></td>
</tr>
<tr>
<td>argumentCollection</td>
<td>Optional</td>
</tr>
<tr>
<td>component</td>
<td>Required</td>
</tr>
<tr>
<td>input_params</td>
<td>Optional</td>
</tr>
<tr>
<td>method</td>
<td>Required</td>
</tr>
<tr>
<td>password</td>
<td>Ignored</td>
</tr>
<tr>
<td>proxyPassword</td>
<td>Invalid</td>
</tr>
<tr>
<td>proxyPort</td>
<td>Invalid</td>
</tr>
<tr>
<td>proxyServer</td>
<td>Invalid</td>
</tr>
<tr>
<td>proxyUser</td>
<td>Invalid</td>
</tr>
<tr>
<td>returnVariable</td>
<td>Optional</td>
</tr>
<tr>
<td><strong>Syntax 3</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>Syntax 4A</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>Syntax 4B</strong></td>
<td>Optional</td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
If the component attribute specifies a component name, the component with the corresponding name is instantiated, the requested method is invoked, and then the component instance is immediately destroyed. If the attribute contains a reference to an instantiated component object, no instantiation or destruction of the component occurs. On UNIX systems, ColdFusion searches first for a file with a name that matches the specified component name, but is all lower case. If it does not find the file, it looks for a file name that matches the component name exactly, with the identical character casing. Method arguments can be passed in any of the following ways. If an argument is passed in more than one way with the same name, this order of precedence applies:

1. Using the cfinvokeargument tag
2. Passing directly as attributes of the cfinvoke tag (they cannot have the same name as a registered cfinvoke attribute: method, component, webservice, returnVariable)
3. Passing as struct keys, using the argumentCollection attribute
   For example, the params struct contains three keys: a=1, b=1, c=1. The following call is evaluated as if the arguments were passed to the method in the order a=3, b=2, c=1:

   ```cfinvoke ... a=2 b=2 argumentCollection==params>
   <cfinvokeargument name="a" value="3">
   </cfinvoke>
   ```

   ![Note]
   The following cfinvoke tag attribute names are reserved; they cannot be used for argument names: component, method, argumentCollection, and result.

Example 1

This example uses Syntax 1.
<!--- Immediate instantiation and destruction. --->
<cfinvoke
    component="nasdaq.quote"
    method="getLastTradePrice"
    returnVariable="res">
    <cfinvokeargument
        name="symbol"
        value="macr">
    </cfinvoke>
    <cfoutput>#res#</cfoutput>
</cfinvoke>

Example2

This example uses Syntax 1.

<!--- Passing the arguments using argumentCollection. --->
<cfset args = StructNew()>
<cfset args.symbol = "macr">
<cfinvoke
    component="nasdaq.quote"
    method="getLastTradePrice"
    argumentCollection="#args#"
    returnVariable="res">
    <cfoutput>#res#</cfoutput>
</cfinvoke>

Example3

This example uses Syntax 2.

<!--- Called only from within a component, MyComponent. --->
<cfinvoke
    method = "a method name of MyComponent"
    returnVariable = "variable name">

Example4

This example uses Syntax 3.

<!--- Using cfinvoke to consume a web service using a ColdFusion component. --->
<cfinvoke
    webservice="http://www.xmethods.net/sd/2001/TemperatureService.wsdl"
    method="getTemp"
    returnvariable="aTemp">
    <cfinvokeargument name="zipcode" value="55987"/>
</cfinvoke>
<cfoutput>The temperature at zip code 55987 is #aTemp#</cfoutput>
For more information on web services, see Using Web Services in the Developing ColdFusion Applications.

Example 5

This example uses Syntax 4A.

```coldfusion
<!--- Separate instantiation and method invocation; useful for multiple invocations using different methods or values. --->
<cfobject
    name="quoteService"
    component="nasdaq.quote">
<cfinvoke
    component="#quoteService#"
    method="getLastTradePrice"
    symbol="macr"
    returnVariable="res_macr">
<cfoutput>#res_macr#</cfoutput>
<cfoutput>#res#</cfoutput>
<cfinvoke
    component="#quoteService#"
    method="getLastTradePrice"
    symbol="mot"
    returnVariable="res_mot">
<cfoutput>#res_mot#</cfoutput>
<cfoutput>#res#</cfoutput>
```
cfinvokeargument

Description

Passes the name and value of a parameter to a component method or a web service. This tag is used in the cfinvoke tag.

Category

Extensibility tags

Syntax

```xml
<cfinvokeargument
    name="argument name"
    value="argument value"
    omit = "yes|no">
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfargument, cfcomponent, cffunction, cfinvoke, cfobject, cfproperty, cfreturn

History

ColdFusion MX 7: Added the omit attribute. ColdFusion MX: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Argument name.</td>
</tr>
<tr>
<td>value</td>
<td>Required</td>
<td></td>
<td>Argument value.</td>
</tr>
<tr>
<td>omit</td>
<td>Optional</td>
<td>no</td>
<td>Enables you to omit a parameter when invoking a web service. It is an error to specify omit=&quot;yes&quot; if the cfinvoke webservice attribute is not specified. yes: omit this parameter when invoking a web service. no: do not omit this parameter when invoking a web service.</td>
</tr>
</tbody>
</table>
Usage

You can have multiple cfinvokeargument tags in a cfinvoke tag body. You can use cfinvokeargument tag to dynamically determine the arguments to be passed. For example, you can use conditional processing to determine the argument name, or you can use a cfif tag to determine whether to execute the cfinvokeargument tag. If you are invoking a web service, you can omit a parameter by setting the omit attribute to "yes". If the WSDL specifies that the argument is nillable, ColdFusion MX sets the associated argument to null. If the WSDL specifies minOccurs=0, ColdFusion omits the argument from the WSDL.

Example1

```
<cfinvoke
    component="nasdaq.quote"
    method="getLastTradePrice"
    returnVariable="res">
    <cfinvokeargument name="symbol" value="mot">
    <cfinvokeargument name="symbol" value="macr">
  </cfinvoke>

  <cfoutput>#res#</cfoutput>

Example2

```
<cfinput src="http://www.xmethods.net/sd/2001/TemperatureService.wsdl" method="getTemp"
      returnvariable="aTemp">
    <cfinvokeargument name="zipcode" value="55987"/>
  </cfinvoke>

  <cfoutput>The temperature at zip code 55987 is #aTemp#</cfoutput>
Tags j-l

This section lists tags starting with alphabets J-L with their description.

- cflayout
- cflayoutarea
- cfldap
- cflocation
- cflock
- cflog
- cflogin
- cfloginuser
- cflogout
- cfloop
- cfloop: index loop
- cfloop: conditional loop
- cfloop: looping over a date or time range
- cfloop: looping over a query
- cfloop: looping over a list, a file, or an array
- cfloop: looping over a COM collection or structure
cflayout

Description

Creates a region of its container (such as the browser window or a cflayoutarea tag) with a specific layout behavior: a bordered area, a horizontal or vertically arranged box, or a tabbed navigator.

Category

Display management tags

Syntax

```
<cflayout
type="accordion|border|hbox|tab|vbox"
activeOnTop="false|true"
align="center|justify|left|right"
fillHeight="true|false"
fitToWindow="true|false"
height="integer"
name="string"
padding="integer"
style="CSS style specification"
tabHeight="measurement"
tabPosition="top|bottom"
tabStrip="true|false"
titleCollapse="true|false"
width="integer">
  cflayoutarea tags
</cflayout>
```

⚠️ Note

You can specify this tag's attribute in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute name as structure key.

See also

cfajaximport, cfdiv, cflayoutarea, cfpod, cfwindow, Using Ajax User Interface Components and Features in the Developing ColdFusion Applications

History

ColdFusion 9:

- Added accordion value of the type attribute and the activeOnTop, fillHeight, and titleCollapse attributes.
- The attributes, height and width are supported for the types hbox and vbox.

ColdFusion 8: Added this tag.

Attributes
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Applies to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Required</td>
<td></td>
<td>all</td>
<td>The type of layout. The following values are valid:</td>
</tr>
</tbody>
</table>
• **accordion**: a container with multiple layout areas that display no more than one area at a time. Each layout area has a title bar that is always displayed. The user can expand or collapse each layout area by clicking + or - button on the layout area title bar.

• **border**: a box with a border and up to five layout areas, each with a border. For more information, see Usage.

• **hbox**: a horizontal box where all immediate child controls are arranged horizontally.

• **tab**: a tabbed display where the current child `cflayoutarea` tag occupies the display area of the layout, and each layout area has a tab that the user can select to display its contents.

• **vbox**: a vertical box where all immediate child controls are arranged vertically.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activeOnTop</td>
<td>Optional</td>
<td>false</td>
<td>accordion</td>
<td>Specifies whether the active panel moves to the top of the layout, becoming the first panel.</td>
</tr>
</tbody>
</table>
| align          | Optional | Determined by browser layout direction | all         | Specifies the default alignment of the content of child layout areas. Each cflayoutarea tag can specify an alignment attribute to override this value. The following values are valid:  
  - center  
  - justify  
  - left  
  - right |
| fillHeight     | Optional | true    | accordion    | A Boolean value that specifies whether to adjust the active layout area's height to fill the available space in the layout control container:  
  - true  
  - false  
  If you specify fillHeight as false, then ColdFusion sets overflow to hidden. |
| fitToWindow | Optional | false | border | A Boolean value that specifies whether the border layout must occupy 100% of the width and height of the window:  
- `true`
- `false`
  
The underlying implementation uses a viewport, hence, any content outside the layout is not accommodated in the layout. When using nested border layouts, if you specify `fitToWindow` as `true`, the layouts nested within the border layout may not automatically fit into the available space. Hence, for nested layouts, it is recommended to use `width` or `size` instead of `fitToWindow`. |
<table>
<thead>
<tr>
<th>height</th>
<th>Optional</th>
<th>600 for border layout; autoheight for others</th>
<th>all</th>
<th>Height of the layout in pixels. For the tab layout, the <code>height</code> attribute has the same functionality as the <code>tabheight</code> attribute. If you specify both <code>height</code> and <code>tabheight</code> attributes, <code>height</code> takes priority over <code>tabheight</code>. The height value specified here takes priority over the height value specified using the <code>style</code> attribute.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Optional</td>
<td>all</td>
<td></td>
<td>The name of the layout region. Should be unique on a page.</td>
</tr>
</tbody>
</table>
| padding | Optional | 0                                           | hbox, vbox | • For `hbox` layouts, specifies the padding on the right side of each child layout area.  
• For `vbox` layouts, specifies the padding at the bottom of each child layout area. You can use any valid CSS length or percent format, such as 10, 10%, 10px, or 10em, for this attribute. The padding is included in the child layout area and takes the style of the layout area. |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>style</td>
<td>Optional</td>
<td>all</td>
<td>A CSS style specification that defines layout styles.</td>
</tr>
<tr>
<td>tabHeight</td>
<td>Optional</td>
<td>tab</td>
<td>Specifies the height of the content area of all child layout areas. You can override this setting by specifying a height setting in a individual <code>cflayoutarea</code> tag style attributes.</td>
</tr>
<tr>
<td>tabPosition</td>
<td>Optional</td>
<td>top</td>
<td>tab</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specifies the location of the tabs relative to the tab region contents.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>bottom</strong>: the tabs appear at the bottom of the layout.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>top</strong>: the tabs appear at the top of the layout.</td>
</tr>
<tr>
<td>tabStrip</td>
<td>Optional</td>
<td>true</td>
<td>tab</td>
</tr>
<tr>
<td>titleCollapse</td>
<td>Optional</td>
<td>true</td>
<td>accordion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specifies whether clicking anywhere on each layout area's title bar expands and collapses the layout area. If <code>false</code>, the user must click the title bar +/- button to expand or collapse a layout area.</td>
</tr>
</tbody>
</table>
width | Optional | all | Width of the layout in pixels. This value takes priority over the width defined using the style attribute. If no value is specified, autoWidth is applied and therefore, content fills the entire screen.

Usage

The immediate children of a `cflayout` tag must be `cflayoutarea` tags or nondisplay tags whose bodies contain one or more `cflayoutarea` tags at the top level. For example, a `cflayout` tag could have a tag such as `cfloop` or `cfquery` as a child, and these tags would have `cflayoutarea` tags in their bodies.

The `border` type layout has the following characteristics:

- The layout control and each of its immediate layout area children is surrounded by a border.
- The control can have up to five children positioned at the left, right, center, top, and bottom of the layout.
- You can configure the child layout areas, except for the center area, to have splitters so that users can expand and collapse them or close them completely.
- The center child layout area occupies all available space in the layout that is not used by any of the other layout areas.
- To specify layout height, use the `height` setting of the `style` attribute.

⚠️ Note

If you specify a border layout on a page that has a `DOCTYPE` declaration, the layout cannot properly determine its height and you must specify the height in a `cflayout` tag `style` attribute.

You can use the following JavaScript functions to access the underlying Ext JS - JavaScript Library objects for border and tab type `cflayout` controls.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ColdFusion.Layout.getBorderLayout</code></td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified border type <code>cflayout</code> control.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.getTabLayout</code></td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified tab type <code>cflayout</code> control.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.getAccordionLayout</code></td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified accordion type <code>cflayout</code> control.</td>
</tr>
</tbody>
</table>

For more information on configuring layout areas, see `cflayoutarea`.

Example
The following example shows a set of nested layouts. The outer layout is a vbox, with two layout areas. The top layout area has a border layout, the bottom layout area contains a form with buttons to control the display of the border layout areas.
<html>
<head>
</head>
<body>
<cflayout name="outerlayout" type="vbox">
<cflayoutarea style="height:400;">
<cflayout name="thelayout" type="border">
<!---- The 100% height style ensures that the background color fills the area. --->
<cflayoutarea position="top" size="100" splitter="true"
style="background-color:##00FFFF; height:100%">
This is text in layout area 1: top
</cflayoutarea>
<cflayoutarea title="Left layout area" position="left"
closable="true" collapsible="true" name="left" splitter="true"
style="background-color:##FF00FF; height:100%">
This is text in layout area 2: left<br />
You can close and collapse this area.
</cflayoutarea>
<cflayoutarea position="center"
style="background-color:##FFFF00; height:100%">
This is text in layout area 3: center<br />
</cflayoutarea>
<cflayoutarea position="right" collapsible="true"
title="Right Layout Area" initcollapsed="true"
style="background-color:##FF00FF; height:100%">
This is text in layout area 4: right<br />
You can collapse this, but not close it.<br />
It is initially collapsed.
</cflayoutarea>
<cflayoutarea position="bottom" size="100" splitter="true"
style="background-color:##00FFFF; height:100%">
This is text in layout area 5: bottom
</cflayoutarea>
</cflayout>
</cflayoutarea>
<cflayoutarea style="height:100; ; background-color:##FFCCFF">
<h3>Change the state of Area 2</h3>
<form>
<cfinput name="expand2" width="100" value="Expand Area 2" type="button"
onClick="ColdFusion.Layout.expandArea('thelayout', 'left');">
<cfinput name="collapse2" width="100" value="Collapse Area 2" type="button"
onClick="ColdFusion.Layout.collapseArea('thelayout', 'left');">
<cfinput name="show2" width="100" value="Show Area 2" type="button"
onClick="ColdFusion.Layout.showArea('thelayout', 'left');">
<cfinput name="hide2" width="100" value="Hide Area 2" type="button"
onClick="ColdFusion.Layout.hideArea('thelayout', 'left');">
</form>
</cflayoutarea>
</cflayout>
</body>
</html>
cflayoutarea

Description

Defines a region within a cflayout tag body, such as an individual tab of a tabbed layout.

Category

Display management tags

Syntax

```
In a border layout
<cflayoutarea
  required
  position="bottom|center|left|right|top"
  optional
  align="left|center|justify|right"
  collapsible="false|true"
  initcollapsed="false|true"
  inithide="false|true"
  maxSize="number of pixels"
  minSize="number of pixels"
  name="string"
  onBindError = "JavaScript function name"
  overflow = "auto|hidden"
  size="number of pixels"
  source="URL"
  splitter="false|true"
  style="CSS style specification"
  title="string">
  area elements
</cflayoutarea>

In a hbox or vbox layout
<cflayoutarea
  optional
  name="string"
  onBindError = "JavaScript function name"
  overflow = "auto|hidden|scroll|visible"
  size="number of pixels"
  source="URL"
  style="CSS style specification">
  area elements
</cflayoutarea>

In a tab layout
<cflayoutarea
  optional
  bindonload="false|true"
  closable="false|true"
  disabled="false|true"
  inithide="false|true"
  name="string"
```
In an accordion layout
<cflayoutarea
    optional
    bindonload="false|true"
    closable="false|true"
    name="string"
    onBindError = "JavaScript function name"
    overflow = "auto|hidden|scroll|visible"
    refreshOnActivate = "false|true"
    selected="false|true"
    source="URL"
    style="CSS style specification"
    title="string"
    titleicon="icon location">

    area elements
</layoutarea>
area elements

If you specify a `source` attribute, all child tags are ignored. If you do not have child tags, close the tag with `/>`.

⚠️ **Note**

You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.

**See also**

cfdiv, cflayout, cfpod, cfwindow, [Ajax JavaScript Functions](#), [Using layouts](#) in [Controlling Ajax user interface](#)
layout in the [Developing ColdFusion Applications](#)

**History**

ColdFusion 8: Added this tag

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Applies to</th>
<th>Description</th>
</tr>
</thead>
</table>
| `align`   | Optional| The cflayout tag `align` attribute value | all | Specifies how to align child controls within the layout area. The following values are valid:  
  • center  
  • justify  
  • left  
  • right |
<p>| <code>bindLoad</code>| Optional| true    | tab, accordion | A Boolean value that specifies whether to execute the source attribute expression when the layout area is first loaded. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>closable</td>
<td>Optional</td>
<td>false</td>
<td>A Boolean value that specifies whether the area can close. Specifying this attribute adds an x icon on the tab or title bar that a user can click to close the area.</td>
</tr>
<tr>
<td>collapsible</td>
<td>Optional</td>
<td>false</td>
<td>A Boolean value that specifies whether the area can collapse. Specifying this attribute adds a &gt;&gt; or &lt;&lt; icon on the title bar that a user can click to collapse the area. You cannot use this attribute for border layout areas with a position attribute value of center.</td>
</tr>
<tr>
<td>disabled</td>
<td>Optional</td>
<td>false</td>
<td>A Boolean value that specifies whether the tab is disabled, that is, whether a user can select the tab to display its contents. Disabled tabs are greyed out. Ignored if the selected attribute value is true.</td>
</tr>
<tr>
<td>initCollapsed</td>
<td>Optional</td>
<td>false</td>
<td>A Boolean value that specifies whether the area is initially collapsed. You cannot use this attribute for border layout areas with a position attribute value of center. Ignored if the collapsible attribute value is false.</td>
</tr>
<tr>
<td>initHide</td>
<td>Optional</td>
<td>false</td>
<td>border, tab, accordion</td>
</tr>
<tr>
<td>maxSize</td>
<td>Optional</td>
<td>-1 (no maximum size)</td>
<td>border</td>
</tr>
<tr>
<td>attribute</td>
<td>type</td>
<td>value/option</td>
<td>layout area (position)</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>----------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>minSize</td>
<td>Optional</td>
<td>-1 (no minimum size)</td>
<td>if the position attribute value is <code>top</code> or <code>bottom</code>, the minimum height of the area, in pixels, that you can set by dragging a splitter. If the position attribute value is <code>left</code> or <code>right</code>, the minimum width of the area. You cannot use this attribute for border layout areas with a position attribute value of <code>center</code>.</td>
</tr>
<tr>
<td>name</td>
<td>Optional</td>
<td>all</td>
<td>The name of the layout area.</td>
</tr>
<tr>
<td>onBindError</td>
<td>Optional</td>
<td>See Description</td>
<td>The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and have specified a global error handler (by using the <code>ColdFusion.setGlobalErrorHandler</code> function), it displays the error message; otherwise a default error pop-up displays.</td>
</tr>
<tr>
<td>overflow</td>
<td>Optional</td>
<td>auto</td>
<td>all</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For accordion, if fillheight attribute of the cflayout tag is false, default value is hidden.</td>
<td>Specifies how to display child content whose size would cause the control to overflow the window boundaries. The following values are valid:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• auto: show scroll bars when necessary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• hidden: do not allow access to overflowing content.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• scroll: always show horizontal and vertical scroll bars, even if they are not needed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• visible: content can display outside the bounds of the layout area.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Notes:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• You cannot use visible or scroll for layout areas in border layouts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• In Internet Explorer, layout areas with the visible setting expand to fit the size of the contents, rather than having the contents extend beyond the layout area.</td>
</tr>
<tr>
<td>position</td>
<td>Required if the cf layout type is border</td>
<td>border</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Required if the cf layout type is border</td>
<td>The position of the area in the layout. Must be one of the following values:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• top: Position the area across the top of the full layout.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• bottom: Position the area across the bottom of the full layout.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• left: Position the area on the left side of the layout, between any visible top and bottom areas.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• right: Position the area on the right side of the layout, between any visible top and bottom areas.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• center: Position the area in the space not taken by the top, bottom, left, and right areas. Border style layouts can have at most one layout area of each type.</td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Value</td>
<td>Tab, Accordion</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| refreshOnActivate | Optional   | false          | Tab, Accordion | • **true**: Refresh the contents of the tab/accordion panel by running the **source** bind expression whenever the tab/panel display region shows (for example, when the user selects the tab), in addition to when bind events occur.  
  • **false**: Refresh the tab/accordion panel display region only when the bind expression is triggered by its bind event. To use this attribute, also specify a **source** attribute. |
<p>| selected          | Optional   | first tab is selected | Tab, Accordion | A Boolean value that specifies whether this tab is initially selected so that its contents appears in the layout.                                |</p>
<table>
<thead>
<tr>
<th>size</th>
<th>Optional</th>
<th><code>-1</code> calculate initial size dynamically</th>
<th>border, hbox, vbox</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>For hbox layouts and border layouts with <code>position</code> attribute values of <code>top</code> or <code>bottom</code>, the initial height of the area. For vbox layouts and border layouts with <code>position</code> attribute values of <code>left</code> or <code>right</code>, the initial width of the area. For hbox and vbox layouts, you can use any valid CSS length or percent format (such as <code>10</code>, <code>10%</code> <code>10px</code>, or <code>10em</code>) for this attribute. For border layouts, this attribute value must be an integer number of pixels. You cannot use this attribute for border layout areas with a <code>position</code> attribute value of <code>center</code>. ColdFusion automatically determines the center size based on the size of all other layout areas. <strong>Note:</strong> If a layout area in a border layout contains only AJAX controls such as HTML format cf tags, specify a <code>size</code> attribute. Otherwise, the AJAX components may not be visible until the layout area is resized.</td>
</tr>
</tbody>
</table>
| source | Optional | all | A URL that returns the layout area contents. ColdFusion uses standard page path resolution rules. You can use a bind expression with dependencies in this attribute. If a file specified in this attribute includes tags that use AJAX features, such as cfgrid, use the cfajaximport tag on the page that includes the cflayout tag. For more information, see cfajaximport.

For more information on the source attribute, see Usage. |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>splitter</td>
<td>Optional</td>
<td>false</td>
<td>A Boolean value that specifies whether the layout area has a divider between it and the adjacent layout area control. Users can drag the splitter to change the relative sizes of the areas. If this attribute is set to true on a left or right position layout area, the splitter resizes the area and its adjacent area horizontally. If this attribute is set to true on a top or bottom position layout area, the splitter resizes the layout vertically. You cannot use this attribute for border layout areas with a position attribute value of center.</td>
</tr>
<tr>
<td>style</td>
<td>Optional</td>
<td>all</td>
<td>A CSS style specification that controls the appearance of the area.</td>
</tr>
<tr>
<td>tabTip</td>
<td>Optional</td>
<td>tab</td>
<td>If defined, a tab tip is displayed.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Usage</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>Optional; required for tab layouts</td>
<td>For tab layouts, the text to display on the tab. For border layouts, if you specify this attribute, ColdFusion creates a title bar for the layout area with the specified text as the title. By default, border layouts that are not closable or collapsible do not have a title bar. You cannot use this attribute for border layout areas with a position attribute value of center.</td>
<td></td>
</tr>
<tr>
<td>{{title}}icon</td>
<td>Optional</td>
<td>accordion</td>
<td>Specifies the location of the icon to display with the title.</td>
</tr>
</tbody>
</table>

**Usage**

All `cflayoutarea` tags must be children of `cflayout` tags and cannot have `cflayoutarea` tags as immediate children, but they can contain `cflayout` tags. However, the `cflayoutarea` tags do not have to be direct children of the `cflayout` tag; instead, the `cflayout` tag could have a tag such as `cfloop` or `cfquery` as a child, and the `cflayoutarea` tags could be in the body of the `cfloop` or `cfquery` tag. These rules let you create arbitrarily complex combinations of different layouts.

**Note**

You cannot put a layout of type `border` inside a layout of type `tab`.

If you do not specify a `size` attribute value, ColdFusion attempts to determine the required size for the layout area contents. However, in some cases, such as when the layout area contains AJAX controls, ColdFusion might not be able to determine the required size, and you must specify the `size` attribute to make the AJAX control appear. In these cases, a scroll bar appears for the layout area.

You can use a `source` attribute or a tag body to specify the layout area contents; if you specify both, ColdFusion uses the contents specified by the `source` attribute and ignores the tag body. If you use a `source` attribute, an animated icon and the text "Loading..." appears while the contents is being fetched.

If the `source` attribute specifies a page that defines JavaScript functions, the function definitions on that page must have the following format:

```
functionName = function(arguments) {function body}
```
Function definitions that use the following format may not work:

```javascript
function functionName (arguments) {function body}
```

However, Adobe recommends that you include all custom JavaScript in external JavaScript files and import them on the application's main page, and not write them inline in code that you get using the source attribute. Imported pages do not have this function definition format restriction.

If you use the `source` attribute, you can use a `bind expression` to include form field values or other form control attributes as part of the source specification. You can bind to HTML format form controls only. For detailed information on using bind expressions see Using Ajax Data and Development Features in the Developing ColdFusion Applications. In border type layouts, a center layout area always takes up any space that is not used by the other areas, even if you do not specify a `cflayoutarea` tag with a `center position attribute`. Therefore, if you want only two layout areas in either direction, one of the two must be the center area, or you must explicitly size the two areas to take up the full layout area.

When you nest layouts, set the inner layout area initial sizes appropriately to ensure that they appear. Use the following JavaScript functions to enable, disable, show, hide, expand, collapse, and select layout areas:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColdFusion.Layout.createTab</td>
<td>Creates a tab in an existing tabbed layout.</td>
</tr>
<tr>
<td>ColdFusion.Layout.disableTab</td>
<td>Disables the specified tab so it cannot be selected.</td>
</tr>
<tr>
<td>ColdFusion.Layout.enableTab</td>
<td>Enables the specified tab so users can select it and display the area contents.</td>
</tr>
<tr>
<td>ColdFusion.Layout.hideTab</td>
<td>Hides a tab.</td>
</tr>
<tr>
<td>ColdFusion.Layout.selectTab</td>
<td>Selects a tab and displays the layout area contents.</td>
</tr>
<tr>
<td>ColdFusion.Layout.showTab</td>
<td>Shows a tab that was hidden using the <code>inithide</code> attribute or the <code>hideTab()</code> function.</td>
</tr>
<tr>
<td>ColdFusion.Layout.collapseArea</td>
<td>Collapses an area of a border layout.</td>
</tr>
<tr>
<td>ColdFusion.Layout.expandArea</td>
<td>Expands a collapsed area of a border layout.</td>
</tr>
<tr>
<td>ColdFusion.Layout.getTabLayout</td>
<td>Hides an area of a border layout.</td>
</tr>
<tr>
<td>ColdFusion.Layout.hideArea</td>
<td>Hides an area of a border layout.</td>
</tr>
<tr>
<td>ColdFusion.Layout.showArea</td>
<td>Shows an area of a border layout that was hidden using the <code>inithide</code> attribute or the <code>hideArea()</code> function.</td>
</tr>
<tr>
<td>ColdFusion.Layout.hideAccordion</td>
<td>Hides an accordion.</td>
</tr>
</tbody>
</table>
ColdFusion.Layout.showAccordion  Shows an accordion that was hidden using the initHide attribute or the hideArea() function.

ColdFusion.Layout.selectAccordion  Selects an accordion and displays the layout area contents.

ColdFusion.Layout.collapseAccordion  Collapses an area of an accordion layout.

ColdFusion.Layout.expandAccordion  Expands a collapsed area of an accordion layout.

**Note**

When you use the style attribute to specify the background color of a border layout area, specify a height style of 100% to make the background color cover the entire layout area. This is because the style specification applies to an inner content area of the layout area, not the layout area itself, and the 100% specification ensures that the content area takes up all available space in the layout area.

**Example**

The following example creates a three-tabbed layout and lets you use buttons to dynamically control the second tab.
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
<h3>Tab layout</h3>
<cflayout type="tab" name="thelayout" tabheight="175" style="background-color:##CCffFF; color:red; height:200">
  <cflayoutarea title="Tab 1" style="background-color:##FFAAFF;" closable="true">
    This is text in layout area 1
  </cflayoutarea>
  <cflayoutarea name="area2" title="Tab 2" inithide="true" style="background-color:##FFCCFF">
    This is text in layout area 2
  </cflayoutarea>
  <cflayoutarea title="Tab 3" style="background-color:##FF99FF;">
    This is text in layout area 3
  </cflayoutarea>
</cflayout>
<br />
<cfform>
  <cfinput name="show" width="40" value="show tab" type="button"
    onClick="ColdFusion.Layout.showTab('thelayout', 'area2');">
  <cfinput name="hide" width="40" value="hide tab" type="button"
    onClick="ColdFusion.Layout.hideTab('thelayout', 'area2');">
  <cfinput name="enable" width="40" value="enable tab" type="button"
    onClick="ColdFusion.Layout.enableTab('thelayout', 'area2');">
  <cfinput name="disable" width="40" value="disable tab" type="button"
    onClick="ColdFusion.Layout.disableTab('thelayout', 'area2');">
  <cfinput name="select" width="40" value="select tab" type="button"
    onClick="ColdFusion.Layout.selectTab('thelayout', 'area2');">
</cfform>
</body>
</html>
**cfldap**

**Description**

Provides an interface to a Lightweight Directory Access Protocol (LDAP) directory server, such as the Netscape Directory Server.

**Category**

Internet protocol tags

**Syntax**

```cml
cfldap
  action = "action"
  server = "server name"
  attributes = "attribute, attribute"
  delimiter = "delimiter character"
  dn = "distinguished name"
  filter = "filter"
  maxRows = "number"
  modifyType = "replace|add|delete"
  name = "name"
  password = "password"
  port = "port number"
  rebind = "yes|no"
  referral = "number of allowed hops"
  returnAsBinary = "column name, column name"
  scope = "scope"
  secure = "multifield security string"
  separator = "separator character"
  sort = "attribute[, attribute]..."
  sortControl = "nocase|desc|asc"
  start = "distinguished name"
  startRow = "row number"
  timeout = "milliseconds"
  username = "user name"
  clientcert = "path to client certificate"
  clientcertpassword = "password for the client certificate"
  usetls = "true|false"
</cfldap>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfftp, cfhttp, cfmail, cfmailparam, cfpop, Managing LDAP Directories in the Developing ColdFusion Applications

**History**
ColdFusion 11: Added 3 new attributes: clientcert, clientcertpassword, and usetls. Removed the filterFile attribute.

ColdFusion 8: Added the ability to use a comma as a delimiter when specifying a list of variables in the returnAsBinary attribute, for example, returnAsBinary="objectGUID,objectSID". Previously, the allowed delimiter was a space.

ColdFusion MX 7: Added the returnAsBinary attribute. Added SSL V2 client based authentication; this means that ColdFusion supports the CFSSL_CLIENT_AUTH option. If CFSSL_CLIENT_AUTH is selected, ColdFusion assumes that the first certificate in the cacerts (or the certificate database) contains the Client Certificate.

ColdFusion MX:

- Changed the name attribute behavior: this tag validates the query name in the name attribute.
- Changed sorting behavior: this tag does not support client-side sorting of query results. (It supports server-side sorting; use the sort and sortcontrol attributes.)
- Changed how results are sorted: server-side sorting results might be sorted slightly differently than in ColdFusion 5. If you attempt a sort against a server that does not support it, ColdFusion MX throws an error.
- Deprecated the filterConfig and filterFile attributes. They might not work, and might cause an error, in later releases.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td>query</td>
<td>• query: returns LDAP entry information only. Requires name, start, and attributes attributes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• add: adds LDAP entries to LDAP server. Requires attributes attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• modify: modifies LDAP entries, except distinguished name dn attribute, on LDAP server. Requires dn. See modifyType attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• modifyDN: modifies distinguished name attribute for LDAP entries on LDAP server. Requires dn.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• delete: deletes LDAP entries on an LDAP server. Requires dn.</td>
</tr>
<tr>
<td>server</td>
<td>Required</td>
<td>Host name or IP address of LDAP server.</td>
<td></td>
</tr>
<tr>
<td>attributes</td>
<td><strong>Required if action = &quot;Query&quot;, &quot;Add&quot;, &quot;ModifyDN&quot;, or &quot;Modify&quot;</strong></td>
<td>For queries: comma-delimited list of attributes to return. For queries, to get all attributes, specify &quot;*&quot;. If action = &quot;add&quot; or &quot;modify&quot;, you can specify a list of update columns. Separate attributes with a semicolon. If action = &quot;ModifyDN&quot;, ColdFusion passes attributes to the LDAP server without syntax checking.</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>delimiter</td>
<td>Optional</td>
<td>; (semicolon)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Separator between attribute name-value pairs. Use this attribute if either of these situations exist:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The attributes attribute specifies more than one item.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• An attribute contains the default delimiter (semicolon), for example: mgrpmsgrejecttext;lang-en</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used by query, add, and modify actions, and by cfldap to output multi-value attributes. For example, if $ (dollar sign), you could specify &quot;cn = Double Tree Inn$street = 1111 Elm; Suite 100, where the semicolon is part of the street value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dn</td>
<td><strong>Required if action = &quot;Add&quot;, &quot;Modify&quot;, &quot;ModifyDN&quot;, or &quot;delete&quot;</strong></td>
<td>Distinguished name, for update action, for example, &quot;cn = Bob Jensen, o = Ace Industry , c = US&quot;</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>filter</td>
<td>Optional</td>
<td>&quot;objectclass = *&quot;</td>
<td>Search criteria for action = &quot;query&quot;. List attributes in the form: &quot;(attribute operator value)&quot;. For example: &quot;(sn = Smith)&quot;</td>
</tr>
<tr>
<td>maxRows</td>
<td>Optional</td>
<td></td>
<td>Maximum number of entries for LDAP queries.</td>
</tr>
</tbody>
</table>
| modifyType| Optional | replace   | How to process an attribute in a multi-value list:  
|           |          |           | - add: appends it to any attributes.  
|           |          |           | - delete: deletes it from the set of attributes.  
|           |          |           | - replace: replaces it with specified attributes. You cannot add an attribute that is already present or that is empty. |
| name      | Required if action = "Query" |           | Name of LDAP query. The tag validates the value. |
| password  | Required if secure = "CF SSL_BASIC" |           | Password that corresponds to user name. If secure = "CF SSL_BASIC", V2 encrypts the password before transmission. |
| port      | Optional | 389       | Port. |
| rebind    | Optional | no        |  
|           |          |           | - yes: attempts to rebind referral callback and reissue query by referred address using original credentials.  
<p>|           |          |           | - no: referred connections are anonymous. |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>referral</td>
<td>Optional</td>
<td>Integer. Number of hops allowed in a referral. A value of 0 disables referred addresses for LDAP; no data is returned.</td>
</tr>
<tr>
<td>returnAsBinary</td>
<td>Optional</td>
<td>A space-delimited list of columns that are to be returned as binary values.</td>
</tr>
<tr>
<td>scope</td>
<td>Optional</td>
<td><strong>oneLevel</strong> Scope of search, from entry specified in start atribute for action = &quot;Query&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• oneLevel: entries one level below entry.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• base: only the entry.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• subtree: entry and all levels below it.</td>
</tr>
<tr>
<td>secure</td>
<td>Optional</td>
<td>Security to employ, and required information. If you specify this attribute, its value must be one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CFSSL_BASIC - Provides V2 SSL encryption and server authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CFSSL_CLIENT_AUTH - This is mandatory if client certificate based authentication is to be used with CFLDAP tag. This is to be used with clientcert and clientcertpassword.</td>
</tr>
<tr>
<td><strong>separator</strong></td>
<td><strong>Optional</strong></td>
<td><strong>, (comma)</strong></td>
</tr>
<tr>
<td><strong>sort</strong></td>
<td><strong>Optional</strong></td>
<td></td>
</tr>
<tr>
<td><strong>sortControl</strong></td>
<td><strong>Optional</strong></td>
<td><strong>asc</strong></td>
</tr>
<tr>
<td><strong>start</strong></td>
<td><strong>Required if action = &quot;Query&quot;</strong></td>
<td></td>
</tr>
<tr>
<td><strong>startRow</strong></td>
<td><strong>Optional</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>timeout</strong></td>
<td><strong>Optional</strong></td>
<td><strong>60000</strong></td>
</tr>
<tr>
<td><strong>username</strong></td>
<td><strong>Required if secure = &quot;CFSSL_BASIC&quot;</strong></td>
<td><strong>(anonymous)</strong></td>
</tr>
</tbody>
</table>
clientcert | Optional | The full path to the key store file that contains the client certificate.
clientcertpassword | Optional | Password for the client certificate.
usetls | Optional | true/false | Whether to use the startTLS extension for initiating SSL over normal LDAP port.

Usage

If you use the query action, cfldap creates a query object, allowing access to information in the query variables, as follows:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryname.recordCount</td>
<td>Number of records returned by query</td>
</tr>
<tr>
<td>queryname.currentRow</td>
<td>Current row of query that cfoutput is processing</td>
</tr>
<tr>
<td>queryname.columnList</td>
<td>Column names in query</td>
</tr>
</tbody>
</table>

If you use the security="CFSSL_BASIC" option, ColdFusion determines whether to trust the server by comparing the server's certificate with the information in the jre/lib/security/cacerts keystore of the JRE used by ColdFusion. The ColdFusion default cacerts file contains information about many certificate granting authorities. If you must update the file with additional information, you can use the keytool utility in the ColdFusion jre/bin directory to import certificates that are in X.509 format. For example, enter the following:

```
keytool -import -keystore cacerts -alias ldap -file ldap.crt -keypass bl19mq
```

Then restart ColdFusion. The keytool utility initial keypass password is "change it". For more information on using the keytool utility, see the Sun JDK documentation. Characters that are illegal in ColdFusion can be used in LDAP attribute names. As a result, the cfldap tag could create columns in the query result set whose names contain illegal characters and are, therefore, inaccessible in CFML. In ColdFusion, illegal characters are automatically mapped to the underscore character; therefore, column names in the query result set might not exactly match the names of the LDAP attributes.

For usage examples, see the Developing ColdFusion Applications.

Example
<h3>cfldap Example</h3>

Provides an interface to LDAP directory servers. The example uses the University of Connecticut public LDAP server. For more public LDAP servers, see <a href="http://www.emailman.com">http://www.emailman.com</a>.

Enter a name and search the public LDAP resource. An asterisk before or after the name acts as a wildcard.

<!---- If form.name exists, the form was submitted; run the query. --->
<cfif IsDefined("form.name")>
<!---- Check to see that there is a name listed. --->
<cfif form.name is not ">
<!---- Make the LDAP query. --->
<cfldap
    server = "ldap.uconn.edu"
    action = "query"
    name = "results"
    start = "dc=uconn,dc=edu"
    filter = "cn=#name#"
    attributes = "cn,o,title,mail,telephonenumber"
    sort = "cn ASC">
<!---- Display results. --->
<center>
<table border = 0 cellspacing = 2 cellpadding = 2>
<tr>
<th colspan = 5>
<cfoutput>#results.recordCount# matches found </cfoutput></TH>
</tr>
<tr>
<th><font size = "-2">Name</font></TH>
<th><font size = "-2">Organization</font></TH>
<th><font size = "-2">Title</font></TH>
<th><font size = "-2">E-Mail</font></TH>
<th><font size = "-2">Phone</font></TH>
</tr>
<cfoutput query = "results">
<tr>
<td><font size = "-2">#cn#</font></td>
<td><font size = "-2">#o#</font></td>
<td><font size = "-2">#title#</font></td>
<td><font size = "-2"><A href = "mailto:#mail#">#mail#</A></font></td>
<td><font size = "-2">#telephonenumber#</font></td>
</tr>
</cfoutput>
</table>
</center>
</cfif>
</cfif>
<form action="#cgi.script_name#" method="POST">
Enter a name to search in the database.
<input type="Text" name="name" />
<input type="Submit" value="Search" name="" />
</form>
**cflocation**

**Description**

Stops execution of the current page and opens a ColdFusion page or HTML file.

**Category**

Flow-control tags, Page processing tags

**Syntax**

```xml
<cflocation
    url = "URL"
    addToken = "yes|no"
    statusCode = "300|301|302|303|304|305|307">
</cflocation>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfabort, cfbreak, cfexecute, cfexit, cfif, cfloop, cfswitch, cfthrow, cftry

**History**

ColdFusion 11: The addToken attribute is modified. If Secure Profile is enabled, addToken's default value will be "false".

ColdFusion 8: Added the `statusCode` attribute.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Required</td>
<td></td>
<td>URL of HTML file or CFML page to open.</td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
| **addToken** | **Optional** | **Default value is false (when the secure profile enabled).** | **The `clientManagement` attribute must be enabled (see [cfapplication](#)).**

- **yes**: appends client variable information to URL.
- **no**

New in ColdFusion 11: If Secure Profile is enabled, addToken's default value will be false. |
<table>
<thead>
<tr>
<th>statusCode</th>
<th>Optional</th>
<th>302</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>The HTTP status code, as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 300 HTTP_MULTIPLE_CHOICES: The requested address refers to more than one entity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 301 HTTP_MOVED_PERMANENTLY: The page is assigned a new URI. The change is permanent.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 302 HTTP_MOVED_TEMPORARILY: The page is assigned a new URI. The change is temporary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 303 HTTPSEE_OTHER: The client should try another network address.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 304 HTTP_NOT_MODIFIED: The requested resource has not been modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 305 HTTP_USE_PROXY: The requested resource must be accessed through the proxy given by the Location field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 307 HTTP_TEMPORARY_REDIRECT: The requested data temporarily resides at a new location. The status codes from 304 to 307 do not redirect you to the page specified in a URL, unless you also follow the guidelines specified in the most recent HTTP RFC.</td>
</tr>
</tbody>
</table>

Usage
You might write a standard message or response in a file, and call it from several applications. Use this tag to redirect the user’s browser to the standard file. This tag has no effect if you use it after the `cfflush` tag on a page.

⚠️ **Note**

When using `cfabort`, `cflocation`, or `cfcontent` tags, the `OnAbort` method is invoked instead of `OnRequestEnd`.

**Example**

```html
<h3>cflocation Example</h3>
<p>This tag redirects the browser to a web resource; normally, you would use this tag to go to a CF page or an HTML file on the same server. The `addToken` attribute lets you send client information to the target page.</p>
<p>If you remove the comments, this code redirects you to CFDOCS home page:</p>

```html
<!--- <cflocation url = "http://localhost:8500/cfdocs/dochome.htm" addToken = "no"> --->
```
**cflock**

**Description**

Ensures the integrity of shared data. Instantiates the following kinds of locks:

- **Exclusive** - Allows single-thread access to the CFML constructs in its body. The tag body can be executed by one request at a time. No other requests can start executing code within the tag while a request has an exclusive lock. ColdFusion issues exclusive locks on a first-come, first-served basis.
- **Read-only** - Allows multiple requests to access CFML constructs within the tag body concurrently. Use a read-only lock only when shared data is read and not modified. If another request has an exclusive lock on shared data, the new request waits for the exclusive lock to be released.

**Category**

*Application framework tags*

**Syntax**

```xml
<cflock
  timeout = "time-out in seconds"
  name = "lock name"
  scope = "Application|Server|Session|Request"
  throwOnTimeout = "yes|no"
  type = "readOnly|exclusive">
  <!--- CFML to be synchronized. --->
</cflock>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfapplication, cfassociate, cfmessagebox, Using Persistent Data and Locking in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added Request value to scope attribute.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeout</td>
<td>Required</td>
<td>Maximum length of time, in seconds, to wait to obtain a lock. If lock is obtained, tag execution continues. Otherwise, behavior depends on the value of the <code>throwOnTimeout</code> attribute. If you set <code>timeout=0</code>, the timeout is determined by the &quot;Timeout Requests after x&quot; setting in the ColdFusion Administrator Settings page, if that setting is enabled. However, if the setting is not enabled, and you set <code>timeout=0</code>, ColdFusion can wait indefinitely to obtain the lock.</td>
</tr>
<tr>
<td>name</td>
<td>Optional</td>
<td>Locks name. Mutually exclusive with the <code>scope</code> attribute. Only one request can execute the code within a <code>cflock</code> tag with a given name at a time. Cannot be an empty string. Permits synchronizing access to resources from different parts of an application. Lock names are global to a ColdFusion server. They are shared among applications and user sessions, but not clustered servers.</td>
</tr>
</tbody>
</table>
### cflock Tag

**scope**
- **Optional**
- **Locks scope. Mutually exclusive with the `name` attribute. Only one request in the specified scope can execute the code within this tag (or within any other `cflock` tag with the same lock scope) at a time.**
  - Application
  - Request
  - Server
  - Session

**throwOnTimeout**
- **Optional**
- **yes**
- **How time-out conditions are handled:**
  - **yes**: exception is generated for the time-out.
  - **no**: execution continues past this tag.

**type**
- **Optional**
- **exclusive**
- **readOnly**: lets more than one request read shared data.
- **exclusive**: lets one request read or write shared data.

---

⚠️ **Note**

Limit the scope of code that updates shared data structures, files, and CFXs. Exclusive locks are required to ensure the integrity of updates, but read-only locks are faster. In a performance-sensitive application, substitute read-only locks for exclusive locks where possible, for example, when reading shared data.

---

**Usage**

ColdFusion is a multi-threaded server; it can process multiple page requests at a time. Use the `cflock` tag for these purposes:

- To ensure that modifications to shared data and objects made in concurrently executing requests occur sequentially.
- Around file manipulation constructs, to ensure that file updates do not fail because files are open for writing by other applications or tags.
- Around CFX invocations, to ensure that ColdFusion can safely invoke CFXs that are not implemented in a thread-safe manner. (This applies only to CFXs developed in C.) To work safely with ColdFusion, a C++ CFX that maintains and manipulates shared (global) data structures must be made thread-safe; however, this
requires advanced knowledge. You can use a CFML custom tag wrapper around a CFX to make its invocation thread-safe. When you display, set, or update variables in a shared scope, use the `scope` attribute to identify the scope as Server, Application or Session.

**Deadlocks**

A **deadlock** is a state in which no request can execute the locked section of a page. After a deadlock occurs, neither user can break it, because all requests to the protected section of the page are blocked until the deadlock can be resolved by a lock time-out.

The `cflock` tag uses kernel level synchronization objects that are released automatically upon time out and/or the abnormal termination of the thread that owns them. Therefore, while processing a `cflock` tag, ColdFusion never deadlocks for an infinite period. However, large time-outs can block request threads for long periods, and radically decrease throughput. To prevent this, always use the minimum time-out value. Another cause of blocked request threads is inconsistent nesting of `cflock` tags and inconsistent naming of locks. If you nest locks, everyone accessing the locked variables must consistently nest `cflock` tags in the same order. Otherwise, a deadlock can occur. These examples show situations that cause deadlocks:

<table>
<thead>
<tr>
<th>Example deadlock with two users</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User 1</strong></td>
</tr>
<tr>
<td>Locks the session scope.</td>
</tr>
<tr>
<td>Deadlock: Tries to lock the Application scope, but it is already locked by User 2.</td>
</tr>
</tbody>
</table>

The following deadlock could occur if you tried to nest an exclusive lock inside a read lock:

<table>
<thead>
<tr>
<th>Example deadlock with one user</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User 1</strong></td>
</tr>
<tr>
<td>Locks the Session scope with a read lock.</td>
</tr>
<tr>
<td>Attempts to lock the Session scope with an exclusive lock.</td>
</tr>
<tr>
<td>Deadlock: Cannot lock the Session scope with an exclusive lock because the scope is already locked for reading.</td>
</tr>
</tbody>
</table>

The following code shows this scenario:

```cfm
<cflock timeout = "60" scope = "SESSION" type = "readOnly">
    ................
<cflock timeout = "60" scope = "SESSION" type = "Exclusive">
    ...........
</cflock>
</cflock>
```

To avoid a deadlock, everyone who nests locks must do so in a well-specified order and name the locks consistently. If you must lock access to the Server, Application, and Session scopes, do so in this order:
1. Lock the Session scope. In the `cflock` tag, specify `scope = "session"`.
2. Lock the Application scope. In the `cflock` tag, specify `scope = "Application"`.
3. Lock the Server scope. In the `cflock` tag, specify `scope = "server"`.
4. Unlock the Server scope.
5. Unlock the Application scope.
6. Unlock the Session scope.

**Note**

If you do not have to lock a scope, you can skip any pair of these lock/unlock steps. For example, if you do not have to lock the Server scope, you can skip Steps 3 and 4. Similar rules apply for named locks.

For more information, see the following:

- [Using Persistent Data and Locking](#) in the *Developing ColdFusion Applications*.
- *Locking thread data and resource access* in the *Developing ColdFusion Applications* (for information on locking the Request scope when you use the `cfthread` tag to create multithreaded ColdFusion applications).
- [ColdFusion Locking Best Practices](#), on the Adobe website.

**Example**

```coldfusion
<!----
This example shows how `cflock` can guarantee consistency of data updates to variables in the Application, Server, and Session scopes. --->

<!---- Copy the following code into an Application.cfm file in the application root directory. --->
<!---------------- Beginning of Application.cfm code ---------------->
<!---- cfapplication defines scoping for a ColdFusion application and enables or disables storing of application and session variables. Put this tag in a special file called Application.cfm. It is run before any other ColdFusion page in its directory. --->

<!---- Enable session management for this application. --->
<cfapplication name = "ETurtle"
    sessionTimeout = #CreateTimeSpan(0,0, 0, 60)#
    sessionManagement = "yes">

<!---- Initialize session and application variables used by E-Turtleneck. Use session scope for the session variables. --->
<cflock scope = "Session"
    timeout = "30" type = "Exclusive">
    <cfif NOT IsDefined("session.size")>
    <cfset session.size = ">
    </cfif>
    <cfif NOT IsDefined("session.color")>
    <cfset session.color = ">
    </cfiff>
</cflock>

<!---------------- End of Application.cfm code -------------------->
```
<!--- Use an application lock for the application-wide variable that keeps track of the number of turtlenecks sold. For a more efficient, but more complex, way of handling Application scope locking, see the "Developing ColdFusion Applications"--->
<cflock scope = "Application" timeout = "30" type = "Exclusive">
  <cfif NOT IsDefined("application.number")>
    <cfset application.number = 0>
  </cfif>
</cflock>

<!----------------------- End of Application.cfm ----------------------->

<h3>cflock Example</h3>

<cfif IsDefined("form.submit")>
  <!--- The form has been submitted; process the request. --->
  <cfoutput>
    Thanks for shopping E-Turtleneck. You chose size <b>#form.size#</b>, color <b>#form.color#</b>.\n  </cfoutput>

  <!--- Lock the code that assigns values to session variables. ---->
  <cflock scope = "Session" timeout = "30" type = "Exclusive">
    <cfparam name = session.size Default = #form.size#>
    <cfparam name = session.color Default = #form.color#>
  </cflock>

  <!---- Lock the code that updates the Application scope number of turtlenecks sold. --->
  <cflock scope = "Application" timeout = "30" type = "Exclusive">
    <cfset application.number = application.number + 1>
    <cfoutput>
      E-Turtleneck has now sold #application.number# turtlenecks!
    </cfoutput>
  </cflock>
</cfif>

<cfelse>
  <!--- Show the form only if it has not been submitted. --->
  <cflock scope = "Application" timeout = "30" type = "Readonly">
    <cfoutput>
      E-Turtleneck has sold #application.number# turtlenecks to date.
    </cfoutput>
  </cflock>
</cfelse>

<form method="post" action="cflocktest.cfm">
  <p>Congratulations! You selected the most comfortable turtleneck in the world. Please select color and size.</p>
  <table cellspacing = "2" cellpadding = "2" border = "0">
    <tr>
      <td>Select a color.</td>
    </tr>
    <td><select type = "Text" name = "color">
      <option>red
      <option>white
      <option>blue
      <option>turquoise
      <option>black
      <option>forest green
    </select>
  </table>
</form>
<td>Select a size.</td>
<td>
<select type="Text" name="size">
  <option>XXsmall</option>
  <option>Xsmall</option>
  <option>small</option>
  <option>medium</option>
  <option>large</option>
  <option>Xlarge</option>
</select>
</td>

<tr>
<td>Press Submit when you are finished making your selection.</td>
<td><input type="Submit" name="submit" value="Submit"></td>
</tr>
</form>
</cfif>
**cflog**

**Description**

Writes a message to a log file.

**Category**

Data output tags

**Syntax**

```xml
<cflog
text = "text"
type = "information|warning|error|fatal"
application = "yes|no"
file = "filename"
log = "log type">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

See also

cfcol, cfcontent, cfoutput, cftable, writeLog()

**History**

ColdFusion MX: Deprecated the `thread`, `date`, and `time` attributes. They might not work, and might cause an error, in later releases. (In earlier releases, these attributes determined whether the respective data items were output to the log. In ColdFusion MX, this data is always output.)

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Required</td>
<td></td>
<td>Message text to log.</td>
</tr>
</tbody>
</table>
| application | Optional| yes     | • yes: logs the application name, if it is specified in a `cfapplication` tag or Application.cfc file.  
<p>|             |         |         | • no         |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>file</strong></td>
<td>Optional</td>
<td>Message file. Specify only the main part of the filename. For example, to log to the Testing.log file, specify &quot;Testing&quot;. The file must be located in the default log directory. You cannot specify a directory path. If the file does not exist, it is created automatically, with the extension .log.</td>
</tr>
</tbody>
</table>
| **log** | Optional | If you omit the **file** attribute, writes messages to standard log file. Ignored, if you specify **file** attribute.  
• **Application**: writes to Application.log, normally used for application-specific messages.  
• **Scheduler**: writes to Scheduler.log, normally used to log the execution of scheduled tasks. |
| **type** | Optional | Information | Type (severity) of the message:  
• **Information**  
• **Warning**  
• **Error**  
• **Fatal** |

**Usage**

This tag logs custom messages to standard or custom log files. You can specify a file for the log message or send messages to the default application or scheduler log. The log message can include ColdFusion expressions. Log files must have the extension .log and must be located in the ColdFusion log directory. Log entries are written as comma-delimited lists with these fields:

- **type**
- **thread**
- **date**
- **time**
- **application**
- **textValues** are enclosed in double quotation marks. If you specify **no** for the **application** attribute, the corresponding entry in the list is empty.  
You can disable **cflogtag** execution. For more information, see the ColdFusion Administrator Basic Security...
The following example logs the name of a user that logs on an application. The message is logged to the file myAppLog.log in the ColdFusion log directory. It includes the date, time, and thread ID, but not the application name.

```
<cflog file="myAppLog" application="no"
text="User #Form.username# logged on.">
```

For example, if a user enters "Sang Thornfield" in a form's username field, this entry is added to the myAppLog.log file entry: "Information", "153", "02/28/01", "14:53:40", "User Sang Thornfield logged on."
**cflogin**

**Description**

A container for user login and authentication code. ColdFusion runs the code in this tag if a user is not already logged in. You put code in the tag that authenticates the user and identifies the user with a set of roles. Used with `cfloginuser` tag.

**Category**

*Security tags*

**Syntax**

```xml
<cflogin
    applicationToken = "token"
    cookieDomain = "domain"
    idletimeout = "value"
    allowconcurrent = "true|false"
    usebasicauth = "true|false">
    ...
    <cfloginuser
        name = "name"
        password = "password"
        roles = "roles">
    </cfloginuser>
</cflogin>
```

**See also**

`cfloginuser, cflogout, GetAuthUser, GetUserRoles, IsUserInAnyRole, IsUserInRole, IsUserLoggedin, Securing Applications` in the *Developing ColdFusion Applications*

**History**

ColdFusion 11: Added 2 new attributes allowconcurrent and usebasicauth.

ColdFusion 8: The `applicationToken` attribute lets you specify a unique application identifier for each application, or the same value for multiple applications. ColdFusion MX 6.1: Changed behavior: the `cflogin` variable exists when ColdFusion receives a request with NTLM or Digest (HTTP Negotiated header) authentication information. ColdFusion MX: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>attribute</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>applicationtoken</td>
<td>Optional</td>
<td>The current application name</td>
</tr>
<tr>
<td>cookiedomain</td>
<td>Optional</td>
<td>Domain of the cookie that is used to mark a user as logged in. Use this attribute to enable a user login cookie to work with multiple clustered servers in the same domain.</td>
</tr>
<tr>
<td>idletimeout</td>
<td>Optional</td>
<td>1800 Time interval, in seconds, after which ColdFusion logs off the user.</td>
</tr>
<tr>
<td>allowconcurrent</td>
<td>Optional</td>
<td>true If concurrent login sessions should be allowed.</td>
</tr>
<tr>
<td>usebasicauth</td>
<td>Optional</td>
<td>true If basic authentication should be used to validate the login.</td>
</tr>
</tbody>
</table>

**Usage**

The body of this tag executes only if there is no logged-in user. When using application-based security, you put code in the body of the `cflogin` tag to check the user-provided ID and password against a data source, LDAP directory, or other repository of login identification. The body must include a `cfloginuser` tag to establish the authenticated user's identity in ColdFusion.

You control the data source and are responsible for coding the SQL within the `cflogin` tag; make sure that the associated database has user, password, and role information.

The `cflogin` tag has a built-in `cflogin` structure that contains two variables, `cflogin.name` and `cflogin.password`, if the page is executing in response to any of the following:

- Submission of a form that contains input fields with the names `j_username` and `j_password`.
- A request that uses HTTP Basic authentication and, therefore, includes an Authorization header with the user name and password.
- A request that uses NTLM or Digest authentication. In this case, the username and password are hashed.
using a one-way algorithm in the Authorization header; ColdFusion gets the username from the web server and sets the `cflogin.password` value to the empty string.

You can use these values in the `cflogin` tag body to authenticate the user, and, in the `cfloginuser` tag, to log the user in. The structure is only available in the `cflogin` tag body.

Example

The following example shows a simple authentication. This code is typically in the `Application.cfc` `onRequestStart` method or in the `application.cfm` page.

```cfml
<cflogin>
  <cfif NOT IsDefined("cflogin")>
    <cfinclude template="loginform.cfm">
    <cfabort>
  <cfelse>
    <cfif cflogin.name eq "admin">
      <cfset roles = "user,admin">
    <cfelse>
      <cfset roles = "user">
    </cfif>
  </cfif>
  <cfloginuser name = "#cflogin.name#" password = "#cflogin.password#"
    roles = "#roles#"/>
</cflogin>
```

The following view-only example checks the user ID and password against a data source:

```cfml
<cfquery name="qSecurity" datasource="UserRolesDb">
  SELECT Roles FROM SecurityRoles
  WHERE username=<cfqueryparam value="#cflogin.name#" CFSQLTYPE="CF_SQL_VARCHAR" AND password=<cfqueryparam value="#cflogin.password#" CFSQLTYPE='CF_SQL_VARCHAR' />
</cfquery>

<cfif qSecurity.recordcount gt 0>
  <cfloginuser name = "#cflogin.name#" password = "#cflogin.password#"
    roles = "#trim(qSecurity.Roles)#" />
</cfif>
```
**cfloginuser**

**Description**

Identifies an authenticated user to ColdFusion. Specifies the user ID and roles. Used within a `cflogin` tag.

**Category**

*Security tags*

**Syntax**

```xml
<cfloginuser
    name = "name"
    password = "password"
    roles = "roles">
```

**See also**

`cflogin`, `cflogout`, `GetAuthUser`, `GetUserRoles`, `IsUserInAnyRole`, `IsUserInRole`, `IsUserLoggedIn`, `Securing Applications` in the *Developing ColdFusion Applications*

**History**

ColdFusion MX 6.1: Changed behavior: if the Session scope is enabled, and the `cfapplication` tag `loginStorage` attribute is set to Session, the login remains in effect until the session expires or the user is logged out by the `cflogout` tag.

ColdFusion MX: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>A user name.</td>
</tr>
<tr>
<td>password</td>
<td>Required</td>
<td></td>
<td>A user password.</td>
</tr>
<tr>
<td>roles</td>
<td>Required</td>
<td></td>
<td>A comma-delimited list of role identifiers. ColdFusion processes spaces in a list element as part of the element.</td>
</tr>
</tbody>
</table>

**Usage**

Used inside the `cflogin` tag to identify the authenticated user to ColdFusion. After you call this function, the `GetAuthUser` and `IsUserInRole` return the user name and role information.
Note

By default, the user information is stored as memory-only cookies. The `cfapplication` tag or the `Application.cfc` This.loginStorage variable can specify that login information is stored in the Session scope.

Example

See `cflogin`.
**cflogout**

**Description**

Logs the current user out. Removes knowledge of the user ID, password, and roles from the server. If you do not use this tag, the user is automatically logged out when the session ends.

**Category**

*Security tags*

**Usage**

```
<cflogout session="all|current|others">
```

**See also**

*cflogin, cfloginuser, GetAuthUser, GetUserRole, IsUserInAnyRole, IsUserInRole, IsUserLoggedIn, Securing Applications* in *Developing ColdFusion Applications*

**History**

ColdFusion 11: Added the session attribute.

ColdFusion MX 6.1: Changed behavior: if the Session scope is enabled, a login remains in effect until the session expires or the user is logged out by the *cflogout* tag.

ColdFusion MX: Added this tag.

**Attribute**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>session</td>
<td>When invoked which session should be logged out. This is a string. The possible values are: • all • current • others</td>
<td>current</td>
</tr>
</tbody>
</table>

**Example**
<cflogin>
<cfloginuser
    name = "foo"
    password ="bar"
    roles = "admin">
</cflogin>
<cfoutput>Authorized user: #getAuthUser()#</cfoutput>
<cflogout>
<cfoutput>Authorized user: #getAuthUser()#</cfoutput>
cfloop

Description

Looping is a programming technique that repeats a set of instructions or displays output repeatedly until one or more conditions are met. This tag supports the following types of loops:

- cfloop: index loop
- cfloop: conditional loop
- cfloop: looping over a date or time range
- cfloop: looping over a query
- cfloop: looping over a list, a file, or an array
- cfloop: looping over a COM collection or structure

For more information, see `cfloop` and `cfbreak` and `Populating arrays with data` in the `Developing ColdFusion Applications`.

Category

Flow-control tags
cfloop: index loop

Description

An index loop repeats for a number of times that is determined by a numeric value. An index loop is also known as a FOR loop.

Syntax

```html
<cfloop
    index = "parameter name"
    from = "beginning value"
    to = "ending value"
    step = "increment"
    charset "charset to read in a file">
    HTML or CFML code ...
</cfloop>
```

See also

cfabort, cfbreak, cfcontinue, cfdirectory, cfexecute, cfexit, cfif, cflocation, cfrethrow, cfswitch, cfthrow, cftry; cfloop and cfbreak in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Required</td>
<td></td>
<td>Index value. ColdFusion sets it to the from value and increments or decrements by step value, until it equals the to value.</td>
</tr>
<tr>
<td>from</td>
<td>Required</td>
<td></td>
<td>Beginning value of index.</td>
</tr>
<tr>
<td>to</td>
<td>Required</td>
<td></td>
<td>Ending value of index.</td>
</tr>
<tr>
<td>step</td>
<td>Optional</td>
<td>1</td>
<td>Step by which to increment or decrement the index value.</td>
</tr>
<tr>
<td>charset</td>
<td>optional</td>
<td></td>
<td>Charset to use when reading in a file line-by-line.</td>
</tr>
</tbody>
</table>

Usage

Using anything other than integer values in the from and to attributes of an index loop can produce unexpected results. For example, if you increment through an index loop from 1 to 2, with a step of 0.1, ColdFusion outputs "1,1.1,1.2,...,1.9", but not "2". This is a programming language problem regarding the internal representation of
floating point numbers.

**Note**

The `to` value is evaluated once, when the `cfloop` tag is encountered. Any change to this value within the `loop` block, or within the expression that evaluates to this value, does not affect the number of times the loop is executed.

**Example**

In this example, the code loops five times, displaying the `index` value each time:

```cfml
<cfloop index = "LoopCount" from = "1" to = "5">
  The loop index is <cfoutput>#LoopCount#</cfoutput>.<br>
</cfloop>
```

The output of this loop is as follows:

- The loop index is 1.
- The loop index is 2.
- The loop index is 3.
- The loop index is 4.
- The loop index is 5.

In this example, the code loops four times, displaying the `index` value each time. The value of `j` is decreased by one for each iteration. This does not affect the value of `to`, because it is a copy of `j` that is made before entering the loop.

```cfml
<cfset j = 4>
<cfloop index = "LoopCount" from = "1" to = #j#>
  The loop index is #LoopCount#<cfoutput>.</cfoutput><br>
  <cfset j = j - 1>
</cfloop>
```

The output of this loop is as follows:

- The loop index is 1.
- The loop index is 2.
- The loop index is 3.
- The loop index is 4.

As before, the value of `j` is decremented by one for each iteration, but this does not affect the value of `to`, because its value is a copy of `j` that is made before the loop is entered.

In this example, `step` has the default value, 1. The code decrements the index:
<cfloop index = "LoopCount"
  from = "5"
  to = "1"
  step = "-1">
The loop index is <cfoutput>#LoopCount#</cfoutput>.<br>
</cfloop>

The output of this loop is as follows:

The loop index is 5.
The loop index is 4.
The loop index is 3.
The loop index is 2.
The loop index is 1.
### cfloop: conditional loop

#### Description

A conditional loop iterates over a set of instructions as long as a condition is True. To use this type of loop correctly, the instructions must change the condition every time the loop iterates, until the condition is False. Conditional loops are known as WHILE loops, as in, "loop WHILE this condition is true."

#### Syntax

```html
<cfloop
    condition = "expression">
    ...
</cfloop>
```

#### See also

cfabort, cfbreak, cfcontinue, cfexecute, cffile, cfif, cfinclude, cflocation, cfswitch, cfthrow, cftry; cfloop and cfbreak in the Developing ColdFusion Applications

#### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>Required</td>
<td></td>
<td>Condition that controls the loop.</td>
</tr>
</tbody>
</table>

#### Example

The following example increments CountVar from 1 to 5.

```html
<!---- Set the variable CountVar to 0. ---->
<cfset CountVar = 0>
<!---- Loop until CountVar = 5. ---->
<cfloop condition = "CountVar LESS THAN 5">
    <cfset CountVar = CountVar + 1>
    The loop index is <cfoutput>#CountVar#</cfoutput>.<br>
</cfloop>
```

The output of this loop is as follows:

```
The loop index is 1.
The loop index is 2.
The loop index is 3.
The loop index is 4.
The loop index is 5.
```
cfl_loop: looping over a date or time range

Description

Loops over the date or time range specified by the from and to attributes. By default, the step is 1 day, but you can change the step by creating a timespan.

Syntax

```cfl
<cfloop
  from = "start time"
  to = "end time"
  index = "current value"
  step = "increment">
</cfloop>
```

See also

cfabort, cfbreak, cfcontinue, cfdirectory, cfexecute, cfexit, cfif, cflocation, cfrethrow, cfswitch, cfthrow, cftry; cfl_loop and cfbreak in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>Required</td>
<td></td>
<td>The beginning of the date or time range.</td>
</tr>
<tr>
<td>to</td>
<td>Required</td>
<td></td>
<td>The end of the date or time range.</td>
</tr>
<tr>
<td>index</td>
<td>Required</td>
<td></td>
<td>Index value. ColdFusion sets it to the from value and increments by the step value, until it equals the to value.</td>
</tr>
<tr>
<td>step</td>
<td>Optional</td>
<td>1 day</td>
<td>Step, expressed as a timespan, by which the index increments.</td>
</tr>
</tbody>
</table>

Example

The following example loops from today's date to today's date plus 30 days, stepping by 7 days at a time and displaying the date:
The following example displays the time in 30-minute increments, starting from midnight and ending 23 hours, 59 minutes, and 59 seconds later:

```
<cfset startTime = CreateTime(0,0,0)>
<cfset endTime = CreateTime(23,59,59)>
<cfloop from="#startTime#" to="#endTime#" index="i"
    step="#CreateTimeSpan(0,0,30,0)#">
    <cfoutput>#TimeFormat(i, "hh:mm tt")#<br /></cfoutput>
</cfloop>
```
cfloop: looping over a query

Description

A loop over a query executes for each record in a query record set. The results are similar to those of the cfoutput tag. During each iteration, the columns of the current row are available for output. The cfloop tag loops over tags that cannot be used within a cfoutput tag.

Syntax

```
<cfloop
    query = "query name"
    startRow = "row number"
    endRow = "row number"
    group = "Query column">
</cfloop>
```

See also

cfabort, cfbreak, cfcontinue, cfexecute, cfexit, cfif, cflocation, cfoutput, cfswitch, cfthrow, cftry: For more information, see cfloop and cfbreak in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>Required</td>
<td></td>
<td>Query that controls the loop. When using query attribute, you can now use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dynamic references in addition to string, as shown in the following code:</td>
</tr>
</tbody>
</table>
|             |         |                                                                         | <cfloop
|             |         |                                                                         | query = "#getEmployees()#">                                                |
| startRow    | Optional|                                                                         | First row of query that is included in the loop.                            |
| endRow      | Optional|                                                                         | Last row of query that is included in the loop.                             |
| group | Optional | Query column to use to group sets of records. Eliminates adjacent duplicate rows when data is sorted. Use if you retrieved a recordset ordered on one or more query columns. For example, if a recordset is ordered on "Customer_ID", you can group the output on "Customer_ID". |

Example

```coldfusion
cfquery name = "MessageRecords" dataSource = "cfdocexamples">
    SELECT * FROM Messages
</cfquery>
<cfloop query = "MessageRecords">
    <cfoutput>#Message_ID#</cfoutput><br>
</cfloop>
```

The `cfloop` tag also iterates over a record set with dynamic start and stop points. This gets the next \( n \) sets of records from a query. This example loops from the fifth through the tenth record returned by the MessageRecords query:

```coldfusion
<cfset Start = 5>
<cfset End = 10>
<cfloop query = "MessageRecords"
    startRow = "#Start#"
    endRow = "#End#">
    <cfoutput>#MessageRecords.Message_ID#</cfoutput><br>
</cfloop>
```

The loop stops when there are no more records, or when the current record index is greater than the value of the `endRow` attribute. The following example combines the pages that are returned by a query of a list of page names into one document, using the `cfinclude` tag:

```coldfusion
<cfquery name = "GetTemplate" dataSource = "Library" maxRows = "5">
    SELECT TemplateName
    FROM Templates
</cfquery>
<cfloop query = "GetTemplate">
    <cfinclude template = "#TemplateName#"/>
</cfloop>
```
cfloop: looping over a list, a file, or an array

Description

Looping over a list steps through elements contained in any of these entities:

- A variable
- A value that is returned from an expression
- An array
- A file Looping over a file does not open the entire file in memory.

Syntax

```
<cfloop
  index = "index name"
  array = "array"
  characters = "number of characters"
  delimiters = "item delimiter"
  file = "absolute path and filename"
  charset = "a charset to use when reading the file"
  list = "list items"
  ...
</cfloop>
```

See also

cfabort, cfbreak, cfcontinue, cfexecute, cfexit, cfif, cflocation, cfswitch, cfthrow, cftry, cfloop and cfbreak in the Developing ColdFusion Applications

History

ColdFusion 8: Added the characters, file, and array attributes.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Required</td>
<td></td>
<td>In a list, file, or array loop, the variable to receive the next element.</td>
</tr>
<tr>
<td>list</td>
<td>Required unless you specify a filename in the file attribute</td>
<td>A list, variable, or filename; contains a list.</td>
<td></td>
</tr>
<tr>
<td>array</td>
<td>Optional</td>
<td></td>
<td>An array.</td>
</tr>
<tr>
<td>characters</td>
<td>Optional</td>
<td>The number of characters to read during each iteration of the loop from the file specified in the <code>file</code> attribute. If the value of the <code>characters</code> attribute is more than the number of characters in the file, ColdFusion uses the number of characters in the file.</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>delimiters</td>
<td>Optional</td>
<td>Characters that separate items in list.</td>
<td></td>
</tr>
<tr>
<td>file</td>
<td>Optional</td>
<td>The absolute pathname of the on-disk or in-memory text file to read, one line at a time. This is helpful when reading large text files, because you can reuse the value of the index variable, which contains the current line of the file. When the loop completes, ColdFusion closes the file.</td>
<td></td>
</tr>
<tr>
<td>charset</td>
<td>Optional</td>
<td>A charset to use when reading the file. See <code>&lt;cffile&gt;</code> for a list of common charsets.</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

This loop displays four names:

```cclike
<cfloop index = "ListElement" list = "John,Paul,George,Ringo">
  <cfoutput>#ListElement#</cfoutput><br>
</cfloop>
```

You can put more than one character in the `delimiters` attribute, in any order. For example, this loop processes commas, colons, and slashes as list delimiters:

```cclike
<cfloop index = "ListElement" list = "John/Paul,George::Ringo" delimiters = ",,:/">
  <cfoutput>#ListElement#</cfoutput><br>
</cfloop>
```
ColdFusion skips the second and subsequent consecutive delimiters between list elements. Thus, in the example, the two colons between “George” and “Ringo” are processed as one delimiter.

To loop over each line of a file, use the tag as follows:

```cfhtml
<cfloop file="c:\temp\simplefile.txt" index="line" charset="UTF-8">
  <cfoutput>#line#</cfoutput>
</cfloop>
```

To read a specified number of characters from a text file during each iteration of the loop, use the tag as follows:

```cfhtml
<cfloop file="c:\temp\simplefile.txt" index="chars" characters="12">
  <cfoutput>#chars#</cfoutput>
</cfloop>
```

When you read the following text file, ColdFusion reads 12 characters during each iteration of the loop; the result appears as follows:

<table>
<thead>
<tr>
<th>Text file</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is line 1.</td>
<td>This is line1. This is line 2. This is line 3. This is line 4.</td>
</tr>
<tr>
<td>This is line 2.</td>
<td>This is line 5. This is line 6. This is line 7. This is line 8.</td>
</tr>
<tr>
<td>This is line 3.</td>
<td>This is line 9. This is line 10. This is line 11.</td>
</tr>
<tr>
<td>This is line 4.</td>
<td></td>
</tr>
<tr>
<td>This is line 5.</td>
<td></td>
</tr>
<tr>
<td>This is line 6.</td>
<td></td>
</tr>
<tr>
<td>This is line 7.</td>
<td></td>
</tr>
<tr>
<td>This is line 8.</td>
<td></td>
</tr>
<tr>
<td>This is line 9.</td>
<td></td>
</tr>
<tr>
<td>This is line 10.</td>
<td></td>
</tr>
<tr>
<td>This is line 11.</td>
<td></td>
</tr>
</tbody>
</table>

To loop over an array, you can do the following:

```cfhtml
<cfset x = ["mars", "earth", "venus", "jupiter"]>
<cfloop array="#x#" index="name">
  <cfoutput>#name#</cfoutput>
</cfloop>
```
cfloop: looping over a COM collection or structure

Description

The cfloop collection attribute loops over every object within a COM/DCOM collection object, or every element in a structure:

- A COM/DCOM collection object is a set of similar items referenced as a group. For example, the group of open documents in an application is a collection.
- A structure contains a related set of items, or it can be used as an associative array. Looping is particularly useful when using a structure as an associative array.

In the loop, each item is referenced by the variable name in the item attribute. The loop executes until all items have been accessed.

The collection attribute is used with the item attribute. In the example that follows, item is assigned a variable called file2, so that with each cycle in the cfloop, each item in the collection is referenced. In the cfoutput section, the name property of the file2 item is referenced for display.

For more information, see Integrating COM and CORBA Objects in CFML Applications in the Developing ColdFusion Applications.

Example

This example uses a COM object to output a list of files. In this example, FFunc is a collection of file2 objects.
<cfobject>
class = FileFunctions.files
name = FFunc
action = Create>
<cfset FFunc.Path = "c:\">
<cfset FFunc.Mask = "*.*" >
<cfset FFunc.attributes = 16 >
<cfset x = FFunc.GetFileList()>
<cfloop collection = #FFUNC# item = "file2">
  <cfoutput> #file2.name# <br> </cfoutput>
</cfloop>
</cfobject>

<!---Loop through a structure that is used as an associative array: --->
...

<!--- Create a structure and loop through its contents. --->
<cfset Departments = StructNew()>
<cfset val = StructInsert(Departments, "John ", "Sales ")>
<cfset val = StructInsert(Departments, "Tom ", "Finance ")>
<cfset val = StructInsert(Departments, "Mike ", "Education ")>

<!--- Build a table to display the contents --->
<cfoutput>
<table cellpadding = "2 " cellspacing = "2 ">
<tr>
<td><b>Employee</b></td>
<td><b>Dept.</b></td>
</tr>
<!--- Use item to create the variable person to hold value of key as loop runs. --->
<cfloop collection = #Departments# item = "person ">
<tr>
<td><b>Employee</b></td>
<td><b>Dept.</b></td>
</tr>
</cfloop>
</table>
</cfoutput>
This section lists tags starting with alphabets M-O with their description.

- `cfmail`
- `cfmailparam`
- `cfmailpart`
- `cfmap`
- `cfmapitem`
- `cfmediaplayer`
- `cfmenu`
- `cfmenuitem`
- `cfmessagebox`
- `cfmodule`
- `cfNTauthenticate`
- `cfoauth`
- `cfobject`
- `cfobject: .NET object`
- `cfobject: COM object`
- `cfobject: component object`
- `cfobject: CORBA object`
- `cfobject: Java or EJB object`
- `cfobject: web service object`
- `cfobjectcache`
- `cfoutput`
cfmail

Description

Sends an e-mail message that optionally contains query output, using an SMTP server.

Category

Communications tags, Internet protocol tags

Syntax

```cfmail
  from = "e-mail address"
  to = "comma-delimited list"
  bcc = "comma-delimited list"
  cc = "comma-delimited list"
  charset = "character encoding"
  debug = "yes|no"
  failto = "e-mail address"
  group = "query column"
  groupcasesensitive = "yes|no"
  mailerid = "header id"
  maxrows = "integer"
  mimeattach = "path"
  password = "string"
  port = "integer"
  priority = "integer or string priority level"
  query = "query name"
  remove = "yes|no"
  replyto = "e-mail address"
  server = "SMTP server address"
  spoolenable = "yes|no"
  startrow = "query row number"
  subject = "string"
  timeout = "number of seconds"
  type = "mime type"
  username = "SMTP user ID"
  useSSL = "yes|no"
  useTLS = "yes|no"
  wraptext = "column number"
  sign = "true|false"
  keystore = "location of keystore"
  keystorepassword = "password of keystore"
  keyalias = "alias of key"
  keypassword = "password for private key"
  encrypt "true|false"
  recipientcert = <path to the public key cert>
  encryptionalgorithm = "DES_EDE3_CBC, RC2_CBC, AES128_CBC, AES192_CBC, AES256_CBC"

(Optional) Mail message body and/or cfmailparam tags

  </cfmail>
```
Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfmailparam, cfmailpart, cfpop, cftp, cfhttp, cfldap, Wrap: Using ColdFusion with mail servers in Sending and Receiving E-Mail in the Developing ColdFusion Applications

History

ColdFusion 11: Added the attributes, encrypt, recipientcert, and encryptionalgorithm.

ColdFusion 8.0.1: Added the attribute Remove.

ColdFusion 8: Added priority, useSSL, and useTLS attributes.

ColdFusion MX 7:

• The cfmail tag no longer lets you send multipart mail by embedding the entire MIME-encoded message in the tag body. Use the cfmailpart tag, instead.
• The cfmail tag renders non-proportional fonts proportionately. This is a behavior change from ColdFusion 5. ColdFusion MX 7 uses UTF-8 and sends this in the mail header (Content-Type: text/plain; charset=UTF-8). ColdFusion 5 uses ISO-8859-1 (Latin 1). To avoid this behavior, add the charset="ISO-8859-1" attribute to restore the default ColdFusion 5 encoding. Alternatively, you can change the encoding on the Mail page in the ColdFusion Administrator.

ColdFusion MX 6.1:

• Added the following attributes: charset, failto, replyto, username, password and wraptext.
• Added support for multiple mail servers in the server attribute.
• Added several configuration options to the ColdFusion Administrator Mail Settings page.

ColdFusion MX: Added the SpoolEnable attribute.

ColdFusion 9: Added support for digitally signing the mail. The following are the relevant attributes that are newly added: sign, keystore, keystorepassword, keyalias, keypassword and remove.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bcc</td>
<td>Optional</td>
<td></td>
<td>Addresses to which to copy the message, without listing them in the message header. To specify multiple addresses, separate the addresses with commas.</td>
</tr>
<tr>
<td>cc</td>
<td>Optional</td>
<td>Addresses to which to copy the message. To specify multiple addresses, separate the address with commas.</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>charset</td>
<td>Optional</td>
<td>Character encoding selected in ColdFusion Administrator Mail page; utf-8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Character encoding of the mail message, including the headers. The following list includes commonly used values:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• utf-8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• iso-8859-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• windows-1252</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• us-ascii</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• shift_jis</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• iso-2022-jp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• euc-jp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• euc-kr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• big5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• hz-gb-2312</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• euc-cn</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• utf-16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information on character encodings, see <a href="http://www.w3.org/International/O-charset.html">www.w3.org/International/O-charset.html</a>.</td>
<td></td>
</tr>
<tr>
<td>debug</td>
<td>Optional</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: sends debugging output to standard output. By default, if the console window is unavailable, ColdFusion sends output to cf_root/runtime/logs/coldfusion-out.log on server configurations.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no: does not generate debugging output.</td>
<td></td>
</tr>
<tr>
<td>failto</td>
<td>Optional</td>
<td>Address to which mailing systems must send delivery failure notifications. Sets the mail envelope reverse-path value.</td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
</tbody>
</table>
| from    | Required | E-mail message sender: 

- A static string; for example, "support@max.com"
- A variable; for example, "#GetUser.EMailAddress#". This attribute does not have to be a valid Internet address; it can be any text string without white spaces. |
| to      | Required | Message recipient e-mail addresses: 

- Static address, for example, "support@com".
- Variable that contains an address, for example, "#Form.Email#".
- Name of a query column that contains an address, for example, "#EMail#". An e-mail message is sent for each returned row. To specify multiple addresses, separate the addresses with commas. |
<p>| subject | Required | Message subject. Can be dynamically generated. For example, to send messages that give customers status updates: &quot;Status of Order Number #Order_ID#&quot;. |</p>
<table>
<thead>
<tr>
<th>group</th>
<th>Optional</th>
<th>CurrentRow</th>
<th>Query column to use when you group sets of records to send as a message. For example, to send a set of billing statements to a customer, group on &quot;Customer_ID.&quot; Case-sensitive. Eliminates adjacent duplicates when data is sorted by the specified field.</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupcasesensitive</td>
<td>Optional</td>
<td>No</td>
<td>Boolean. Whether to consider case when using the group attribute. To group on case-sensitive records, set this attribute to Yes.</td>
</tr>
<tr>
<td>keyalias</td>
<td>Optional</td>
<td></td>
<td>Alias of the key with which the certificate and private key are stored in the keystore. If it is not specified, the first entry in the keystore is chosen as the alias.</td>
</tr>
<tr>
<td>keypassword</td>
<td>Optional</td>
<td></td>
<td>Password for your private key. If not specified, the keystorepassword is used.</td>
</tr>
<tr>
<td>keystore</td>
<td>Optional</td>
<td></td>
<td>The location of the keystore file, for example, C:\OpenSSL\bin\keystore.jks.</td>
</tr>
<tr>
<td>keystorepassword</td>
<td>Optional</td>
<td></td>
<td>The password of the keystore. This is stored in ColdFusion configuration files.</td>
</tr>
<tr>
<td>mailerid</td>
<td>Optional</td>
<td>ColdFusion Application Server</td>
<td>Mailer ID to be passed in X-Mailer SMTP header, which identifies the mailer application.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Default</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>maxrows</td>
<td>Optional</td>
<td>Maximum number of messages to send when looping over a query.</td>
<td></td>
</tr>
<tr>
<td>mimeattach</td>
<td>Optional</td>
<td>Path of the on-disk or in-memory file to attach to message. Attached file is MIME-encoded. ColdFusion attempts to determine the MIME type of the file; use the cfnai lparam tag to send an attachment and specify the MIME type.</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td>A password to send to SMTP servers that require authentication. Requires a username attribute.</td>
<td></td>
</tr>
<tr>
<td>port</td>
<td>Optional</td>
<td>TCP/IP port on which SMTP server listens for requests (normally 25). A value here overrides the Administrator.</td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td>Optional</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>priority attributes:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• An integer in the range 1-5; 1 represents the highest priority.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• One of the following string values, which correspond to the numeric values: highest or urgent, high, normal, low, and lowest or non-urgent.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>query</td>
<td>Optional</td>
<td>Name of cfquery from which to draw data for messages. Use this attribute to send more than one message, or to send query results within a message.</td>
<td></td>
</tr>
<tr>
<td>remove</td>
<td>Optional</td>
<td>no</td>
<td>If yes, ColdFusion removes attachment files (if any) after the mail is successfully delivered.</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>----</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>replyto</td>
<td>Optional</td>
<td></td>
<td>Addresses to which the recipient is directed to send replies.</td>
</tr>
<tr>
<td>server</td>
<td>Optional</td>
<td></td>
<td>SMTP server address, or (Enterprise edition only) a comma-delimited list of server addresses, to use for sending messages. At least one server must be specified here or in the ColdFusion Administrator. A value here overrides the Administrator. A value that includes a port specification overrides the port attribute. For details, see Usage.</td>
</tr>
<tr>
<td>sign</td>
<td></td>
<td></td>
<td>Digitally signs the mail. If set to true, all messages that you send will have digital signature.</td>
</tr>
<tr>
<td>Option</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>spoolenable</td>
<td>Optional</td>
<td>Whether to spool mail or always send it immediately. Overrides the ColdFusion Administrator Spool mail messages to disk for delivery setting.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>yes</strong>: saves a copy of the message until the sending operation is complete. Pages that use this option might run slower than the ones that use the <strong>No</strong> option.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>no</strong>: queues the message for sending, without storing a copy until the operation is complete. If a delivery error occurs when this option is <strong>No</strong>, ColdFusion generates an Application exception and logs the error to the mail.log file.</td>
<td></td>
</tr>
<tr>
<td>startrow</td>
<td>Optional</td>
<td>1</td>
<td>Row in a query to start from.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional</td>
<td>Number of seconds to wait before timing out connection to SMTP server. A value here overrides the Administrator.</td>
<td></td>
</tr>
</tbody>
</table>
| **type** | **Optional** | **text/plain** | MIME type of the message. Can be a valid MIME media type or one of the following:
* text: specifies text/plain type.
* plain: specifies text/plain type.
* html: specifies text/html type.
For a list of all registered MIME media types, see [www.iana.org/assignments/media-types/](http://www.iana.org/assignments/media-types/).

| **username** | **Optional** | A user name to send to SMTP servers that require authentication. Requires a password attribute.

| **useSSL** | **Optional** | Whether to use Secure Sockets Layer.

| **useTLS** | **Optional** | Whether to use Transport Level Security.

| **wraptext** | **Optional** | **Do not wrap text** | The maximum line length, in characters of the mail text. If a line has more than the specified number of characters, replaces the last white space character, such as a tab or space, preceding the specified position with a line break. If there are no white space characters, inserts a line break at the specified position. A common value for this attribute is 72.

| **encrypt** | **Optional** | false | If the Email should be encrypted.

| **recipientcert** | **Optional** | false | The path to the public key certificate of the recipient.
encryptionalgorithm | Optional | The encryption algorithm to use. The valid algorithms are:

- DES_EDE3_CBC
- RC2_CBC
- AES128_CBC
- AES192_CBC
- AES256_CBC

The encryption support is provided through S/MIME.

Usage

Sends a mail message to the specified address. Mail messages can include attachments. The tag body can include CFML code to generate mail output. The cfmailparam and cfmailpart tags can only be used in the cfmail tag body. Mail messages can be single or multipart. If you send a multi-part mail message, all message content must be in cfmailpart tags; ColdFusion ignores multipart message text that is not in cfmailpart tags.

⚠️ Note

The cfmail tag does not make copies of attachments when spooling mail to disk. If you use the cfmail tag to send a message with an attachment with spooling enabled and you use the cfif tag to delete the attachment file, ColdFusion might not send the mail because the mailing process might execute after the file was deleted. (When this happens, the mail log includes a FileNotFound exception and the e-mail is not sent.) You can prevent this problem by setting SpoolEnable="No" in the attribute or disabling spooling in the ColdFusion Administrator. Disabling spooling causes the e-mail to be delivered immediately.

If you set type="text", sometimes whitespace might be compressed in the messages that you send. To resolve this, in the ColdFusion Administrator, go to Server Settings > Settings and then deselect the option Enable Whitespace Management.

Mail addressing

Mail addresses can have any of the following forms:

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>user [server]company.com</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:user@server">user@server</a></td>
<td><a href="mailto:rsmith@company.com">rsmith@company.com</a></td>
</tr>
<tr>
<td>DisplayName <a href="mailto:user@server">user@server</a></td>
<td>Rob Smith <a href="mailto:rsmith@company.com">rsmith@company.com</a></td>
</tr>
<tr>
<td>&quot;DisplayName&quot; <a href="mailto:user@server">user@server</a></td>
<td>&quot;Rob Smith&quot; <a href="mailto:rsmith@company.com">rsmith@company.com</a></td>
</tr>
<tr>
<td>user@server (DisplayName)</td>
<td><a href="mailto:rsmith@company.com">rsmith@company.com</a> (Rob Smith)</td>
</tr>
</tbody>
</table>
Specifying mail servers

The server attribute can specify one or more mail servers.

⚠️ Note

If you specify multiple mail servers in ColdFusion Standard, the cfmail tag uses only the first server in the specification. ColdFusion logs a warning message to the mail log file and ignores the additional servers.

For each server, you can optionally specify a user name, password, and port. These values override the corresponding attributes, if any. The server attribute has the following format:

[user:password@]server[:port],[user:password@]server[:port],....

For example, the following line specifies one server, mail.myco.com that uses the default port and no user or password, and a second server with a user, password, and specific port:

```
server=mail.myco.com,mail_admin:adm2qzf@mail2.myco.com:24
```

When you specify multiple mail servers in ColdFusion Enterprise, ColdFusion tries the available servers in the order they are listed until it connects to a server. ColdFusion does not try to connect to a server that was unavailable in the last 60 seconds.

Digital Signature

To add digital signature to your mail, specify the attributes sign, keystore, keystorepassword, keyalias, and keypassword as provided in the following example:

```
<cfmail from="Sender@Company.com" server="sendmail.myCo.com" sign="true"
keystore="C:\OpenSSL\bin\hello.jks" keystorepassword="digital"
to="Recepient@Company.com" keyalias="crypto" keypassword="signature" subject="Mail with Digital Signature">
```

To add digital signature to all the mails you send, instead of adding the attributes to the tag, specify the settings in the Server Settings > Settings page of the ColdFusion Administrator.

If you do not specify the attributes in the tag, the Administrator settings are applied. Also, in the tag, if you set sign = "true" and do not specify the attributes keystore, keystorepassword, keyalias, and keypassword, then the values for these attributes specified using ColdFusion Administrator are applied.

Example
<h3>cfmail Example</h3>

<!--- Delete the surrounding comments to use this example. --->

<cfif IsDefined("form.mailto")>
<cfif form.mailto is not "" AND form.mailfrom is not "" AND form.Subject is not ">
<cfmail to = "#form.mailto#" from = "#form.mailFrom#" subject = "#form.subject#">
This message was sent by an automatic mailer built with cfmail:
#form.body#
</cfmail>
</cfif>
</cfif>

<h3>Thank you</h3>
<p>Thank you, <cfoutput>#mailfrom#: your message, #subject#, has been sent to #mailto#</cfoutput>.</p>

<form action = "cfmail.cfm" method="POST">
<pre>
TO: <input type = "Text" name = "MailTo">
FROM: <input type = "Text" name = "MailFrom">
SUBJECT:  <input type = "Text" name = "Subject">
<hr>
MESSAGE BODY:
<textarea name = "body" cols="40" rows="5" wrap="virtual"></textarea>
</pre>
<!---- Establish required fields. --->
<input type = "hidden" name = "MailTo_required" value = "You must enter a recipient">
<input type = "hidden" name = "MailFrom_required" value = "You must enter a sender">
<input type = "hidden" name = "Subject_required" value = "You must enter a subject">
<input type = "hidden" name = "Body_required" value = "You must enter some text">
<p><input type = "Submit" name = ""></p>
</form>

In ColdFusion 11, you can encrypt your mails using the following procedure:

1. Get the certificate of the user for whom you wanted to send the mail (.p7c format).
2. Specify the path to the generated certificate for the recipientcert attribute. For instance:
   <cfmail ... encrypt="true" ENCRYPTIONALGORITHM = "DES_EDE3_CBC"
   recipientcert="#path#cert.p7c" >
3. Configure the mail server from your ColdFusion Administrator.
4. Download the [Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files](https:// Oracle website for the JRE version that ColdFusion is configured with (if you are using an encryption algorithm whose key size is greater than 128).
5. Send the mail. See the following example.
<cfset path = GetDirectoryFromPath(getCurrentTemplatePath())>

<cfset algorithms = "DES_EDE3_CBC,RC2_CBC,AES128_CBC,AES192_CBC,AES256_CBC">

<cfloop index="algo" list="#algorithms#">
    <cfmail to="joe@work.com" from="bob@work.com"
        subject="Sending a mail with the encryption algo : #algo#"
        sign="false"
        encrypt="true"
        ENCRYPTIONALGORITHM = "#algo#"
        recipientcert="#path#cert.p7c" >
        This message is encrypted using the algorithm : #algo#.
    </cfmail>
</cfloop>
**cfmailparam**

**Description**

Attaches a file or adds a header to an e-mail message.

**Category**

*Communications tags, Internet protocol tags*

**Syntax**

```cfc
<cfmail
   to = "recipient"
   subject = "message subject"
   from = "sender"
   more attributes... >
<cfmailparam
   contentID = "content ID"
   disposition = "disposition type">
   file = "filename"
   type = "media type"
</cfmailparam>

OR

<cfmailparam
   name = "header name"
   value = "header value">
   ...
</cfmailparam>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfmail, cfmailpart, cfftp, cfhttp, cfldap, cfpop; Using the cfmailparam tag in Sending and Receiving E-Mail in the Developing ColdFusion Applications

**History**

ColdFusion 8.0.1: Added the attributes Content and Remove.

ColdFusion MX 6.x: Added the Disposition and ContentID attributes.

ColdFusion MX 6.1: Added the type attribute.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>content</th>
<th>Lets you send the contents of a ColdFusion variable as an attachment. To do so, specify the variable in # signs as the content attribute value, as in the following example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentID</td>
<td>Optional                                                                 The Identifier for the attached file. This ID must be globally unique and is used to identify the file in an IMG or other tag in the mail body that references the file content.</td>
</tr>
<tr>
<td>disposition</td>
<td>Optional attachment How the attached file is to be handled. Can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• attachment: presents the file as an attachment.</td>
</tr>
<tr>
<td></td>
<td>• inline: displays the file contents in the message.</td>
</tr>
<tr>
<td>file</td>
<td>Required if you do not specify name attribute Attaches a file in a message. Mutually exclusive with name attribute. The file is MIME encoded before sending.</td>
</tr>
<tr>
<td>name</td>
<td>Required if you do not specify file attribute Name of header. Case-insensitive. Mutually exclusive with file attribute.</td>
</tr>
<tr>
<td>remove</td>
<td>Optional</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Usage

This tag attaches a file or adds a header to an e-mail message. It can only be used in the `cfmail` tag. You can use multiple `cfmailparam` tags within a `cfmail` tag. You can use this tag to include a file, such as an image, in an HTML mail message. The file can be displayed inline in an HTML message, or as an attachment, as Example 2 shows. To include multiple files, use multiple `cfmailparam` tags.

Display a file inline in a mail message

1. Specify `type="html"` in the `cfmail` tag.
2. Specify `disposition="inline"` and a ContentID attribute in the `cfmailparam` tag.
3. Use a `src="cid:ContentIDValue"` attribute to identify the content to include in the HTML tag such as the `img` tag.

Example

Example 1: This view-only example uses the `cfmailparam` tag to add a header to a message, attach files, and to return a receipt to the sender.
<cfmail from = "peter@domain.com" To = "paul@domain.com"
Subject = "See Important Attachments and Reply">
   Please review the new logo. Tell us what you think.
</cfmail>

Example 2: This view-only example displays an image in the body of an HTML message.

<cfmail type="HTML"
   to = "#form.mailto#"
   from = "#form.mailFrom#"
   subject = "Sample inline image">
   <cfmailparam file="C:\Inetpub\wwwroot\web.gif"
      disposition="inline"
      contentID="image1">
      <p>There should be an image here</p>
      <img src="cid:image1">
      <p>After the picture</p>
   </cfmailparam>
</cfmail>
**cfmailpart**

**Description**

Specifies one part of a multipart e-mail message. Can only be used in the `cfmail` tag. You can use more than one `cfmailpart` tag within a `cfmail` tag.

**Category**

Communications tags, Internet protocol tags

**Syntax**

```html
<cfmail
  ...
  (Optional cfmailparam entries)
  <cfmailpart
    charset="character encoding"
    type="mime type"
    wraptext="number"
  >
  Mail part contents
  </cfmailpart>
  ...
</cfmail>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfmail, cfmailparam, cfpop, cfftp, cfhttp, cfldap, cfcontent, Wrap: E-mail in the Developing ColdFusion Applications

**History**

ColdFusion MX 6.1: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
| charset | Optional | Character encoding specified by `charset` attribute of `cfmail` tag | The character encoding in which the part text is encoded. The following list includes commonly used values:
- utf-8
- iso-8859-1
- windows-1252
- us-ascii
- shift_jis
- iso-2022-jp
- euc-jp
- euc-kr
- big5
- hz-gb-2312
- euc-cn
- utf-16
For more information on character encodings, see [www.w3.org/International/O-charset.html](http://www.w3.org/International/O-charset.html).

| type | Required | The MIME media type of the part. Can be a can be valid MIME media type or one of the following:
- text: specifies text/plain type.
- plain: specifies text/plain type.
- html: specifies text/html type.
Note: For a list of all registered MIME media types, see [www.iana.org/assignments/media-types/](http://www.iana.org/assignments/media-types/). |
wraptext | Optional | Do not wrap text | Specifies the maximum line length, in characters of the mail text. If a line has more than the specified number of characters, replaces the last white space character, such as a tab or space, preceding the specified position with a line break. If there are no white space characters, inserts a line break at the specified position. A common value for this attribute is 72.

Usage

Use this tag to create mail messages with alternative versions of the message that duplicate the content in multiple formats. The most common use is to send a plain text version of the message that can be read by all mail readers followed by a version formatted in HTML for display by HTML-compatible mail readers. Specify the simplest version first, with more complex versions afterwards. For more information, see www.ietf.org/rfc/rfc2046.txt.

Example

```cfnlxml
<h3>cfmailpart Example</h3>
<cfmail from = "peter@domain.com" To = "paul@domain.com"
        Subject = "Which version do you see?">
  <cfmailpart type="text" wraptext="74">
    You are reading this message as plain text, because your mail reader does not handle HTML text.
  </cfmailpart>
  <cfmailpart type="html">
    <h3>HTML Mail Message</h3>
    <p>You are reading this message as <strong>HTML</strong>.</p>
    <p>Your mail reader handles HTML text.</p>
  </cfmailpart>
</cfmail>
```
**cfmap**

**Description**

Embeds a geographical map within a ColdFusion web page. Currently, ColdFusion supports only embedding of Google map. To generate a map, provide a valid Google map API key, and specify the latitude and longitude of the location, or the address of the location. The Google map API key can be specified in the following ways:

1. Using the `cfajaximport` tag. You specify the map API key in the `params` attribute as follows:

   ```cfajaximport params="#{googlemapkey='Map API Key'}#"```

2. Using `Application.cfc` as follows:

   ```cfset this.googlemapkey="Map API Key"```  

3. Using the Settings page in the ColdFusion Administrator. Specify the map API key in the Google Map Key field. You can also specify the map API key in `runtime.cfc`

**Category**

[Display management tags](#)

**Syntax**

```
<cfmap
   centeraddress="address"
   centerlatitude="latitude in degrees"
   centerlongitude="longitude in degrees"
   collapsible="true|false"
   continuouszoom="true|false"
   doubleclickzoom="true|false"
   height="integer"
   hideborder="true|false"
   initshow="true|false"
   markerbind="bind expression"
   markercolor="marker color"
   markericon="icon path"
   markerwindowcontent="content"
   name="name"
   onerror="JavaScript function name"
   onload="JavaScript function name"
   overview="true|false"
   scrollwheelzoom="true|false"
   showallmarkers="true|false"
   showcentermarker="true|false"
   showmarkerwindow="true|false"
   showscale="true|false"
   showUser="true|false"
   tip="center property marker tips"
   title="string"
   type="map|satellite|hybrid|earth|terrain"
   typecontrol="none|basic|advanced"
   width="integer"
   zoomcontrol="none|small|large|small3d|large3d"
   zoomlevel="integer">
</cfmap>

See also

<cfdiv, cfwindow, cfmapitem>

History

ColdFusion 10: Added the attribute showUser}}

ColdFusion 9.0.1: Added the attribute {{initShow.

ColdFusion 9: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>centeraddress</td>
<td>Required if centerlatitude and centerlongitude are not specified</td>
<td></td>
<td>The address of the location, which is set as the center of the map.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------</td>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td>centerlatitude</td>
<td>Required if centeraddress is not specified</td>
<td>The latitude value for the location, in degrees. This value is set as the center of the map. This attribute must be used with the centerlatitude attribute. The valid values for centerlatitude are -90 to +90.</td>
<td></td>
</tr>
<tr>
<td>centerlongitude</td>
<td>Required if centeraddress is not specified</td>
<td>The longitude value for the location, in degrees. This value is set as the center of the map. This attribute must be used with the centerlongitude attribute. The valid values for centerlatitude are -180 to +180.</td>
<td></td>
</tr>
</tbody>
</table>
| collapsible       | Optional                                                         | false                   | Whether to provide a collapsible property for the surrounding panel:  
|                   |                                                                  |                         | • true  
|                   |                                                                  | • false                 | If you set collapsible to true, you cannot set hideborders to true. |
| continuouszoom    | Optional                                                         | true                    | Whether to provide zoom control that enables smooth zooming for the map:  
|                   |                                                                  |                         | • true  
|                   |                                                                  | • false                 | |
| doubleclickzoom   | Optional                                                         | true                    | Whether to enable double-click zoom:  
|                   |                                                                  |                         | • true  
<p>|                   |                                                                  | • false                 | |
| height            | Optional                                                         | 400 pixels              | Height of the map, in pixels. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Optional</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hideborder</td>
<td>Optional</td>
<td>true</td>
<td>Whether to hide border for surrounding panel:&lt;br&gt;• true&lt;br&gt;• false  &lt;br&gt;If you set <code>hideborder</code> to true, you cannot set <code>collapsible</code> to true.</td>
</tr>
<tr>
<td>initshow</td>
<td>Optional</td>
<td>true</td>
<td>Use to show/hide a map when the page loads. This is useful if you have collapsible divs or collapsible spry regions, where the user has to show the map on click of a link or button.</td>
</tr>
<tr>
<td>markerbind</td>
<td>Optional</td>
<td></td>
<td>A bind expression to dynamically populate data in the window that is opened when you click the marker icon. The bind expression can specify a CFC function, a JavaScript function, or a URL.</td>
</tr>
<tr>
<td>markercolor</td>
<td>Optional</td>
<td></td>
<td>The color of the marker specified as a hexadecimal value. By default, the centermarker is green in color. The attributes <code>markercolor</code> and <code>markericon</code> are mutually exclusive.</td>
</tr>
<tr>
<td>markericon</td>
<td>Optional</td>
<td></td>
<td>Location of an image file to use as the marker icon. The attributes <code>markericon</code> and <code>markercolor</code> are mutually exclusive.</td>
</tr>
<tr>
<td>markerwindowcontent</td>
<td>Optional</td>
<td></td>
<td>Static content displayed in the marker window. This attribute is mutually exclusive with the <code>markerbind</code> attribute.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Name of the map. The name attribute is required to invoke JavaScript functions.</td>
</tr>
<tr>
<td>onerror</td>
<td>Optional</td>
<td></td>
<td>The JavaScript function to run when there is a Google map API error. The JavaScript function is passed with two parameters, Google map status code and error message.</td>
</tr>
<tr>
<td>onload</td>
<td>Optional</td>
<td></td>
<td>Custom JavaScript function that runs after the map loads, for instance, registering an event.</td>
</tr>
<tr>
<td>overview</td>
<td>Optional</td>
<td>false</td>
<td>Whether to add an Overview panel to the map:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• false</td>
</tr>
<tr>
<td>showmarkerwindow</td>
<td>Optional</td>
<td>false</td>
<td>If set to true, displays the marker window. If the attribute markerbind is used, unless you set this attribute to true, the marker window is not displayed. This attribute is ignored if markerwindowcontent is set to true.</td>
</tr>
<tr>
<td>Property</td>
<td>Default</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>showUser</td>
<td>false</td>
<td>If set to true, on HTML-compliant browsers, user location is shown on the map. For browsers that are not HTML 5 compliant, the address falls back to the value you specify for centerAddress. If no value is specified, it falls back to the value specified for centerLatitude and centerLongitude. User has to authenticate the site so that it tracks user location. For example, in Google Chrome, you are prompted to Allow to track your Physical location.</td>
<td></td>
</tr>
<tr>
<td>scrollwheelzoom</td>
<td>true</td>
<td>Whether to enable mouse wheel zooming control:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false</td>
<td></td>
</tr>
<tr>
<td>showallmarkers</td>
<td>true</td>
<td>Whether to display all markers added to the map:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>When you specify show allmarkers as true, to display all the markers within the map area, the zoom level specified for the map may be overridden.</td>
<td></td>
</tr>
<tr>
<td>showcentermarker</td>
<td>true</td>
<td>Whether to display the marker icon that identifies the map center:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false</td>
<td></td>
</tr>
<tr>
<td>showscale</td>
<td>false</td>
<td>Whether to show scale control:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• false</td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>tip</td>
<td>Optional</td>
<td></td>
<td>A short description of the center location that appears as a tool tip.</td>
</tr>
<tr>
<td>title</td>
<td>Optional</td>
<td></td>
<td>Title of the panel. You cannot define a title, if you set hideborder to true.</td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
<td>map</td>
<td>Type of the Google map:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>map, satellite, hybrid, terrain, earth: If you use type=&quot;earth&quot;, you are prompted to download Google Earth 3D plug-in.</td>
<td></td>
</tr>
<tr>
<td>typecontrol</td>
<td>Optional</td>
<td>basic</td>
<td>Whether to provide a type control that lets you switch the map:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>basic: Displays the marker types that provides the options map, satellite, and hybrid. none advanced: Displays a drop-down list with five options defined for the attribute type.</td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>Optional</td>
<td>400 pixels</td>
<td>Map width, in pixels.</td>
</tr>
<tr>
<td>zoomcontrol</td>
<td>Optional</td>
<td>small</td>
<td>Whether to enable zoom control:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>none, small, large, large3d, small3d</td>
<td></td>
</tr>
<tr>
<td>zoomlevel</td>
<td>Optional</td>
<td>3</td>
<td>Specifies the starting zoom value.</td>
</tr>
</tbody>
</table>
This tag can be used to create a map within an HTML page, a div tag, or in a new window. If you use this tag in a new window, you must use the cfmap tag within the cfwindow tag.

The {{zoomcontrol}} attribute lets you change the size of the embedded map. You can increase the zoom value to get a close-up view of the map. Or, decrease the zoom value to view a larger area of the map at a reduced size. Each time you change the zoom value, the entire map does not refresh, but only those portions of the map that change, making the display of data fast.

The cfmap tag supports the map display in five formats - map, satellite, terrain, earth, and hybrid. The map format displays a standard road map image. The satellite format displays a satellite image of the map. The hybrid format displays a combination of the roadmap and the satellite image of the map, with important street names and places marked on the satellite image.

The following attributes do not work if type="earth": Zoomlevel, showScale, overview, tip, zoomControl, showCenterMarker, and showAllMarkers.

For cfmap tag to work on Safari 3.x and Google Chrome, specify the HTML head tag (<head></head>).

Examples

```html
<h3>cfmap Example using latitude and longitude attributes</h3>
<cfmap name="gmap01"
  centerlatitude="71.094224"
  centerlongitude="42.339641"
  doubleclickzoom="true"
  overview="true"
  scrollwheelzoom="true"
  showscale="true"
  tip="My Map"
  zoomlevel="4"/>

<h3>cfmap Example using center address</h3>
<cfmap name="gmap02"
  centeraddress="345 Park Avenue, san jose, CA 95110-2704, USA"
  doubleclickzoom="true"
  scrollwheelzoom="true"
  showscale="false"
  tip="My Map"/>
```
**cfmapitem**

**Description**

The `cfmapitem` tag is a child tag of the `cfmap` tag. This tag creates markers on the map. You can specify the marker in a map using either the `cfmapitem` tag or using the `ColdFusion.Map.AddMapMarker` JavaScript API. See `ColdFusion.Map.addMarker` for details.

**Category**

Display management tags

**Syntax**

```
<cfmapitem
   address="address"
   latitude="latitude in degrees"
   longitude="longitude in degrees"
   markercolor="marker color"
   markericon="icon path"
   markerwindowcontent="content"
   name="name of the map"
   showmarkerwindow="true|false"
   showUser="true|false"
   tip="marker tip" />
```

**See also**

cfdiv, cfwindow, cfmap

**History**

ColdFusion 10: Added the attribute `{{showUser}}`

ColdFusion 9: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>Required, if latitude and longitude are not specified</td>
<td></td>
<td>The address of the location to set the map marker.</td>
</tr>
<tr>
<td>latitude</td>
<td>Required, if address is not specified</td>
<td></td>
<td>The latitude value for the marker, in degrees. The valid values for latitude are -90 to + 90.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>longitude</td>
<td>Required, if address is not specified</td>
<td></td>
<td>The longitude value for the marker, in degrees. The valid values for longitude are -180 to + 180.</td>
</tr>
<tr>
<td>markerColor</td>
<td>Optional</td>
<td>green</td>
<td>The color of the marker specified as a hexadecimal value. By default, the centermarker is green in color. The attributes markerIcon and markerColor are mutually exclusive.</td>
</tr>
<tr>
<td>markerIcon</td>
<td>Optional</td>
<td></td>
<td>Location of an image file to use as the marker icon. The attributes markerIcon and markerColor are mutually exclusive.</td>
</tr>
<tr>
<td>markerWindowContent</td>
<td>Optional</td>
<td></td>
<td>Static content displayed in the marker window. This attribute ignores the markerBind attribute defined in the tag cfmap.</td>
</tr>
<tr>
<td>name</td>
<td>Optional</td>
<td></td>
<td>The name of the map.</td>
</tr>
<tr>
<td>showMarkerWindow</td>
<td>Optional</td>
<td>Inherits parent cfmap setting</td>
<td>If set to true, displays the marker window. If the attribute markerBind is used, unless you set this attribute to true, the marker window is not displayed.</td>
</tr>
<tr>
<td>showUser</td>
<td>Optional</td>
<td>false</td>
<td>If set to true, on HTML-compliant browsers, user location is shown on the map.</td>
</tr>
<tr>
<td>tip</td>
<td>Optional</td>
<td></td>
<td>A short description of the marker location that appears as a tool tip.</td>
</tr>
</tbody>
</table>

**Usage**

This tag must be used within the `cfmap` tag.
The following inheritance rules apply:
• The value specified for the attribute `showmarkerwindow` in the `cfmap` tag is inherited by all `cfmapitem` tags.
• A child `cfmapitem` tag can override the attribute `showmarkerwindow` in the `cfmap` tag by changing the value.
• Any bind expression defined using the `markerbind` is ignored if the `cfmapitem` tag defines the attribute `markerwindowcontent`.

Examples

```xml
<h3>cfmapitem example using latitude and longitude attributes</h3>
<cfmap name="gmap01"
   centerlatitude="71.094224"
   centerlongitude="42.339641"
   doubleclickzoom="true"
   overview="true"
   scrollwheelzoom="true"
   showscale="true"
   tip="My Map"
   zoomlevel="4">
   <cfmapitem name="marker01"
      latitude="70.50"
      longitude="42.50"
      tip="New marker"/>
</cfmapitem>

<h3>cfmap Example using address address</h3>
<cfmap name="gmap02"
   centerlatitude="71.094224"
   centerlongitude="42.339641"
   doubleclickzoom="true"
   overview="true"
   scrollwheelzoom="true"
   showscale="true"
   tip="My Map"
   zoomlevel="4">
   <cfmapitem name="marker02"
      address="345 Park Avenue, san jose, CA 95110-2704, USA"
      tip="New marker"/>
</cfmapitem>
```
**cfmediaplayer**

**Description**

Creates an in-built media player that lets you play videos in any format supported by HTML 5 compliant browsers in addition to FLV, MPEG-3, and MPEG-4 files. FLV files can be played from any web server. You can play MPEG-3/MPEG-4 only from Flash Media Server using RTMP. You can also play audio files of MP3 format.

**Category**

**Display management tags**

**Syntax**

```xml
<cfmediaplayer
    align="alignment option"
    autoplay="true|false"
    bgcolor="hexadecimal value"
    hideborder="true|false"
    hidetitle="true|false"
    controlbar="true|false"
    fullScreenControl="yes|no"
    name="name"
    onComplete="JavaScript function name"
    onError="JavaScript function name"
    onPause="JavaScript function name"
    onLoad="JavaScript function name"
    onStart="JavaScript function name"
    posterImage="URL"
    quality="low|high|medium"
    repeat="true|false"
    skin="XML_Path"
    source="source name"
    style="style specification"
    title="title"
    type="html|flash">
    height="integer"
    width="integer"
    wmode="window|opaque|transparent">
</cfmediaplayer>
```

**History**

**ColdFusion 10:** type, repeat, posterImage, title, skin, onPause, {{onError}}

**ColdFusion 9:** Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>align</strong></td>
<td>Optional</td>
<td>left</td>
<td>Specifies the horizontal alignment of the media player. You can select from left, right, and center.</td>
</tr>
</tbody>
</table>
| **autoplay** | Optional | false | A Boolean value that specifies if the media player must automatically play the FLV file on loading the CFM page:  
- true  
- false |
| **bgcolor** | Optional | 6b7c85 | The background color of the media player specified as a Hexadecimal value or or a recognized color name, for example red. In the case of HTML player, this attribute applies only if you are using the attribute wmode set to transparent. The dependency does not exist if you are using Flash player. |
| **controlbar** | Optional | true | A Boolean value that specifies if you want to display the control panel for the media player:  
- true  
- false |
| **hideborder** | Optional | true | A Boolean value that specifies if you want a border for the media player panel:  
- true  
- false |
<p>| <strong>hidetitle</strong> | Optional | true | If true, hides the video file name. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fullScreenControl</td>
<td>Optional</td>
<td>yes</td>
<td>Whether full screen is enabled:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following characteristics apply:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Single click plays/pauses the media player.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Borders, if defined, are not displayed in full-screen mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Double-click for full-screen mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- (If you are in full-screen mode) Press &quot;Esc&quot; or double-click to restore the player to normal mode. This attribute is unsupported for HTML playback. Also, you cannot disable full screen for Flash player.</td>
</tr>
<tr>
<td>height</td>
<td>Optional</td>
<td>275 pixels</td>
<td>Height of the media player, in pixels.</td>
</tr>
<tr>
<td>name</td>
<td>Required if source is not defined</td>
<td></td>
<td>Name of the media player. The name attribute is required when you invoke JavaScript functions.</td>
</tr>
<tr>
<td>onComplete</td>
<td>Optional</td>
<td></td>
<td>Custom JavaScript function to run when the FLV file has finished playing.</td>
</tr>
<tr>
<td>onError</td>
<td></td>
<td></td>
<td>Custom JavaScript function to run when playback results in an error.</td>
</tr>
<tr>
<td>onLoad</td>
<td>Optional</td>
<td></td>
<td>Custom JavaScript function to run on loading of the player component.</td>
</tr>
<tr>
<td>onPause</td>
<td></td>
<td></td>
<td>Custom JavaScript function to run when the video is paused.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>onStart</td>
<td>Optional</td>
<td></td>
<td>Custom JavaScript function to run when the FLV file starts playing.</td>
</tr>
<tr>
<td>posterImage</td>
<td></td>
<td></td>
<td>Sets a poster image for the video playback. Takes URL or relative address as value.</td>
</tr>
<tr>
<td>quality</td>
<td>Optional</td>
<td>high</td>
<td>The quality of the media playback:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• medium</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• high</td>
</tr>
<tr>
<td>repeat</td>
<td></td>
<td>false</td>
<td>If true, continues playback from first to last frame after the media player reaches the end of the video.</td>
</tr>
<tr>
<td>skin</td>
<td></td>
<td></td>
<td>Path to the XML in which skinning options are specified. Applies only to Flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example, <code>&lt;cfmediaplayer source=&quot;myvideo.mp4&quot; skin=&quot;./skin/myskin.xml&quot;&gt;</code></td>
</tr>
<tr>
<td>source</td>
<td>Required if name is not defined</td>
<td></td>
<td>The URL to the FLV file. This can be a URL relative to the current page. You can store the FLV file on the ColdFusion server or any other streaming server.</td>
</tr>
<tr>
<td>style</td>
<td>Optional</td>
<td>The following are the supported styles:</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>---------------------------------------</td>
<td></td>
</tr>
<tr>
<td>bgcolor: The background color of the media player.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>borderbottom: A numeric value. The default is 10.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bordertop: A numeric value. The default is 10.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>borderleft: A numeric value. The default is 10.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>borderright: A numeric value. The default is 10.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>titletextcolor: Hexadecimal value of RGB color. For example, specify the color white as #FFFFFF or FFFFFF. The default is black.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>titlebgcolor: Hexadecimal value of RGB color. The default is black.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>progresscolor: The foreground color for the progress bar. Hexadecimal value of RGB color. The default is black.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>progressbgcolor: The background color for the progress bar. Hexadecimal value of RGB color. The default is black.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>controlscolor: The foreground color of the controls in the Control panel. Hexadecimal value of RGB color. The default is black.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>controlbarbgcolor: The background color of the controls. Hexadecimal value of RGB color. The default is black.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Value</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td></td>
<td>Sets title on the media player. The title appears over the media player on upper-left corner. If title is specified and hideTitle is not specified, then hideTitle is set to false. Also when playback is Flash, the attribute wmode for Flash player is set to opaque, and ignores the default/user-specified value.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>flash</td>
<td>The media player type, if html or flash.</td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>480 pixels</td>
<td>Width of the media player, in pixels.</td>
<td></td>
</tr>
</tbody>
</table>
| wmode     | window | Specifies the absolute positioning and layering capabilities in your browser:  
  - window: Plays the media player in its own rectangular window on a web page  
  - opaque: Hides everything behind the media player on the web page  
  - transparent: Lets the background of the web page show through the transparent portions of the media player  
When you set playback type as Flash, value of this attribute is set to opaque, and ignores the default/user-specified value. |

**Example**

In this example, the FLV file is stored in the web root used by the ColdFusion server. You need to store an FLV file - mediafile.flv in the location _web_root__xyz_. You can now create the media player with the following content:
<h3>cfmediaplayer Example</h3>
<cfmediaplayer
  name="Myvideo"
  source="/xyz/mediafile.flv"
  width=500
  height=400
  align="center"
  quality="high"
  fullscreencontrol="true"/>

The following code illustrates styling of the media player:

```coldfusion
<cfset bgColorTheme = "EDC393">
<cfset titleColorTheme = "800517">
<cfset controlsColorTheme = titleColorTheme>
<cfset progressColorTheme = "E67451">
<cfset progressbgColorTheme = "FFF8C6">
<cfset style="bgcolor:#bgColorTheme#;titletextcolor:#titleColorTheme#;titlebgcolor:#bgColorTheme#;controlbarbgcolor:#bgColorTheme#;controlscolor:#controlsColorTheme#;progressbgcolor:#progressbgColorTheme#;progresscolor:#progressColorTheme#;borderleft:20;borderright:20;bordertop:10;borderbottom:13" hideborder="false" hideTitle=false controlbar="true"
source="#defaultFlvfile#">
```
**cfmenu**

**Description**

Creates a horizontal or vertical menu. Any menu item can be the top level of a submenu.

**Category**

*Display management tags*

**Syntax**

```html
<cfmenu
brcolor="HTML color value"
childStyle="CSS style specification"
font="HTML font family"
fontColor="HTML color value"
fontSize="Number of pixels"
menuStyle="CSS style specification"
name="string"
selectedFontColor="HTML color value"
selectedItemColor="HTML color value"
type="horizontal|vertical"
width="Number of pixels">

cfmenuitem tags

</cfmenu>
```

The `cfmenu` tag must have a body that contains at least one `cfmenuitem` tag to define the menu items and an end `</cfmenu>` tag.

**Note**

You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.

**See also**

`cfajaximport`, `cfmenuitem`, `Using menus and toolbars` in *Using Ajax User Interface*

`Components and Features` in the *Developing ColdFusion Applications*

**History**

ColdFusion 8: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Tag</th>
<th>Optional</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
</table>
| bgColor   | Optional | Background color style of the menu               | The color of the menu background. You can use any valid HTML color specification. This specification has the following behaviors:  
- You can override it locally by specifying the menuStyle attribute of this tag and any cfmenuitem tag.  
- It controls the background of color surrounding a submenu whose background is specified by a childStyle attribute. |
| childStyle| Optional | A CSS style specification that applies to the following menu items:  
- The items of the top-level menu  
- All child menu items, including the children of submenus  
This attribute lets you use a single style specification for all menu items. |
| font      | Optional | Browser default font                              | The font to use for all child menu items. Use any valid HTML font-family style attribute. Some common values are serif, sans-serif, Times, Courier, and Arial. |
| fontColor | Optional | black                                             | The color of the menu text. Use any valid HTML color specification. |
| **fontsize** | Optional | **font size of the menu item** | The size of the font. Use a numeric value, such as 8, to specify a pixel character size. Use a percentage value, such as 80%, to specify a size relative to the default font size. Font sizes larger than 20 pixels can result in submenu text exceeding the menu boundary. |
| **menustyle** | Optional | **A CSS style specification that applies to the menu, including any parts of the menu that do not have items. If you do not specify style information in the cfmnuitem tags, this attribute controls the style of the top-level items.** |
| **name** | Optional | **The name of the menu.** |
| **selectedfontcolor** | Optional | **black** | The color of the text for the menu item that has the focus. Use any valid HTML color specification. |
| **selecteditemcolor** | Optional | **light blue** | The color that highlights the menu item that has the focus. You can use any valid HTML color specification. |
| **type** | Optional | **horizontal** | The orientation of the menu. The following values are valid:  
- **horizontal**: Menu items are arranged horizontally.  
- **vertical**: Menu items are arranged vertically. Submenus of both menu types are always arranged vertically. |
<table>
<thead>
<tr>
<th>width</th>
<th>Optional</th>
<th>Width of the container</th>
<th>The width of a vertical menu; not valid for horizontal menus. Use a numeric value, such as 50, to specify a pixel size. Use a percentage value, such as 30%, to specify a size relative to the parent element's size.</th>
</tr>
</thead>
</table>

Usage

The `cfmenu` tag defines a horizontal or vertical ColdFusion menu. You use a single `cfmenu` tag to define the general menu characteristics, and you use `cfmenuitem` child tags to define the individual menu entries and any submenus. You create submenus by putting `cfmenuitem` tags in the body of a `cfmenuitem` tag. You cannot nest a `cfmenu` tag inside a form or inside a `cfmenu` tag or `cfmenuitem` tag.

Example

The following example creates a simple menu bar. When you click an entry in the bar, the browser displays the Adobe website page for the selected product. You can expand the ColdFusion item by clicking the icon, and then select an item to display a specific ColdFusion web page.

```html
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>
<cfmenu name="menu" type="horizontal" fontsize="14" bgcolor="#CCFFFF">
  <cfmenuitem name="acrobat" href="http://www.adobe.com/acrobat" display="Acrobat"/>
  <cfmenuitem name="aftereffects" href="http://www.adobe.com/aftereffects" display="After Effects"/>
  <!--- The ColdFusion menu item has a pop-up menu. --->
  <cfmenuitem name="coldfusion" href="http://www.adobe.com/products/coldfusion" display="ColdFusion">
    <cfmenuitem name="buy" href="http://www.adobe.com/products/coldfusion/buy/" display="Buy"/>
    <cfmenuitem name="devcenter" href="http://www.adobe.com/devnet/coldfusion/" display="Developer Center"/>
    <cfmenuitem name="documentation" href="http://www.adobe.com/support/documentation/en/coldfusion/" display="Documentation"/>
    <cfmenuitem name="support" href="http://www.adobe.com/support/coldfusion/" display="Support"/>
  </cfmenuitem>
  <cfmenuitem name="flex" href="http://www.adobe.com/flex" display="Flex"/>
</cfmenuitem>
</body>
</html>
```
**cfmenuitem**

**Description**

Defines an entry in a menu, including an item that is the head of a submenu.

**Category**

Display management tags

**Syntax**

```
<cfmenuitem
   display="string"
   childStyle="CSS style specification"
   href="URL or JavaScript function"
   image="path"
   menuStyle="CSS style specification"
   name="string"
   style="CSS style specification"
   target="location identifier">
   Optional child menuitem tags
</cfmenuitem>

OR

<cfmenuitem
   divider[="true"]/>
```

If the `cfmenuitem` tag does not have a body with an end `</cfmenuitem>` tag, close the tag with a forward slash character before the closing greater than character (`/`), for example, `<cfmenuitem divider="true"/>`.

**Note**

You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.

**See also**

`cfmenu`, `Using menus and toolbars` in *Using Ajax User Interface Components and Features* in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td>Required if divider attribute is not specified</td>
<td>'</td>
<td>The text to show as the menu item label.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Required</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>childStyle</td>
<td>Optional</td>
<td>A CSS style specification that applies to all child menu items, including the children of submenus.</td>
<td></td>
</tr>
<tr>
<td>divider</td>
<td>Optional</td>
<td>This attribute specifies that the item is a divider. If you specify this attribute, you cannot specify any other attributes. You can use this attribute without a value, as in the following example:{{&lt;cfmenuitem divider /&gt;.}} You cannot use this attribute in a top-level horizontal menu.</td>
<td></td>
</tr>
<tr>
<td>href</td>
<td>Optional</td>
<td>A URL link to activate or JavaScript function to call when the user clicks the menu item.</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>Optional</td>
<td>URL of an image to display at the left side of the menu item. The file type can be any format that the browser can display. For most displays, you must use 15x15 pixel images, because larger images conflict with the menu item text.</td>
<td></td>
</tr>
<tr>
<td>menuStyle</td>
<td>Optional</td>
<td>A CSS style specification that controls the overall style of any submenu of this menu item. This attribute controls the submenu of the current menu item, but not any child submenus of the submenu.</td>
<td></td>
</tr>
<tr>
<td>style</td>
<td>Optional</td>
<td>A CSS style specification that applies to the current menu item only. It is not overridden by the childStyle attribute.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Optional</td>
<td>The name of the menu item.</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>---------------------------</td>
<td></td>
</tr>
<tr>
<td>target</td>
<td>Optional</td>
<td>The current window and frame (if any)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The target in which to display the contents returned by the <code>href</code> attribute. The attribute can be a browser window or frame name, or an HTML target value, such as <code>self</code>.</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

Every `cfmenuitem` tag must be a child of a `cfmenu` tag or a `cfmenuitem` tag. To create a submenu, put the `cfmenuitem` tags for submenu items in the body of the `cfmenuitem` tag for the submenu root in the parent menu. For an example of a simple submenu, see `cfmenu`.

**Example**

The following menu shows the effects of the various style attributes on the menu and menu item appearance.
cfmessagebox

Description

Defines a control for displaying pop-up messages. The control has more features than the standard alert box, including the ability to include a prompt and entry field in the box.

Category

Display management tags

Syntax

```xml
<cfmessagebox 
   bodyStyle = "CSS style specification"
   buttonType = "yesno|yesnocancel"
   callbackHandler = "function name"
   icon = "error|info|question|warning"
   labelCancel = "Cancel button label text"
   labelNo = "No button label text"
   labelOk = "OK button label text"
   labelYes = "Yes button label text"
   message = "message text"
   modal = "yes|no"
   multiline = "false|true"
   name = "control name"
   title = "title"
   type = "alert|confirm|prompt"
   width = "number of pixels"
   x = "numeric pixel coordinate"
   y = "numeric pixel coordinate"/>
```

History

ColdFusion 9: Added this tag

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodyStyle</td>
<td>Optional</td>
<td></td>
<td>A CSS style specification for the body of the message box. As a general rule, use this attribute to set color and font styles.</td>
</tr>
</tbody>
</table>
| buttonType          | Optional         | yesno | Applies to the control type confirm. The buttons to display on the message box:  
|                    |                 |      | • yesno: displays the buttons Yes and No  
|                    |                 |      | • yesnocancel: displays the buttons Yes, No, and Cancel  
| callbackhandler    | Optional        |      | The function that the control calls when a user clicks one of the buttons. For more information see Usage.  
| icon               | Optional        |      | Specifies the following CSS classes:  
|                    |                 |      | • error: Provides the error icon. You can use this icon when displaying error messages.  
|                    |                 |      | • info: Provides the info icon. You can use this icon when displaying any information.  
|                    |                 |      | • question: Provides the question icon. You can use this icon in a confirmation message box that prompts a user response.  
|                    |                 |      | • warning: Provides the warning icon. You can use this icon when displaying a warning message.  
| labelCancel        | Optional        | Cancel | The text to put on the cancel button of a prompt message box.  
| labelOk            | Optional        | OK    | The text to put on an alert button and prompt message box OK button.  


<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>labelNo</td>
<td>Optional</td>
<td>No</td>
<td>The text to put on the button used for a negative response in a confirm message box.</td>
</tr>
<tr>
<td>labelYes</td>
<td>Optional</td>
<td>Yes</td>
<td>The text to put on the button used for a positive response in a confirm message box.</td>
</tr>
<tr>
<td>message</td>
<td>Optional</td>
<td></td>
<td>The text to display inside the message box.</td>
</tr>
<tr>
<td>modal</td>
<td>Optional</td>
<td>yes</td>
<td>A Boolean value that specifies if the message box must be a modal window:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td>multiline</td>
<td>Optional</td>
<td>false</td>
<td>Valid only for prompt type message boxes. A boolean value specifying whether the prompt input text box has a single or multiple lines for text input.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>The control name. Used to refer to the control in JavaScript.</td>
</tr>
<tr>
<td>title</td>
<td>Optional</td>
<td></td>
<td>The title for the message box. If you do not specify a title, ColdFusion assigns the control type value as the default title.</td>
</tr>
<tr>
<td>attribute</td>
<td>required</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
</tbody>
</table>
| type      | Required | The control type. Must be one of the following:
|           |          | - alert - A message with a single OK button.
|           |          | - confirm - A message box with two buttons YES and NO or three buttons YES, NO, and CANCEL.
|           |          | - prompt - a message box with a single-line or multiline text input area and OK and CANCEL buttons. |
| width     | Optional | Width of the message box in pixels. |
| x         | Optional | The X (horizontal) coordinate of the upper-left corner of the message box. ColdFusion ignores this attribute if you do not set the y attribute. |
| y         | Optional | The Y (vertical) coordinate of the upper-left corner of the message box. ColdFusion ignores this attribute if you do not set the x attribute. |

Usage

The cfmessagebox creates a message box, but does not show it. You show a message box, say named mymessagebox, in JavaScript code as follows:

```
ColdFusion.MessageBox.show("mymessagebox");
```

If you specify a callbackhandler, clicking a button in the message box invokes the callbackhandler by passing the button label as a parameter. For prompt boxes, an additional parameter containing the prompt text is also passed. For alert and confirm boxes:

```
var function_name = function(button);
```

For prompt boxes:
```javascript
var function_name = function(button, promptmessage);
```

The EventObject parameter is the JavaScript ID (not the name) of the button that was pressed. The textmessage parameter is a string with the contents of the prompt text box.

**Example**

The following example has three buttons, one to display each type of message box. The message box labels are customized, and the messagebox callback function displays the type of the clicked button.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Untitled Document</title>
<script type="text/javascript">
//Function to to show result of a message box.
var showResult1 = function(btn,message){
    alert("You entered: "+message);
}
//Function to show results of other message boxes.
var showResult2 = function(btn){
    alert("You clicked button: "+btn);
}
//The button onClick handler displays the message boxes.
function showMB(mbox)  {
    ColdFusion.MessageBox.show(mbox);
}
</script>
</head>
<body>
<form>
<p>Click a button display the corresponding message box.</p>
<input name="Prompt" type="button" value="Prompt" onclick="showMB('mymessagebox01')">
<input name="Prompt" type="button" value="Prompt" onclick="showMB('mymessagebox02')">
<input name="Prompt" type="button" value="Prompt" onclick="showMB('mymessagebox03')">
</form>
<!--- Code to define the message boxes. --->
<cfmessagebox name="mymessagebox01" type="prompt" message="Write a short description about yourself" labelOK="This is OK" labelCANCEL="Cancel this" callbackhandler="showResult1" multiline="true"/>
<cfmessagebox name="mymessagebox02" type="confirm" message="Is it OK to save the planet?" labelNO="Dont Save" labelYES="Sure" callbackhandler="showResult2"/>
<cfmessagebox name="mymessagebox03" type="alert" message="You have been ALERTED!" callbackhandler="showResult2" />
</body>
</html>
**cfmodule**

**Description**
Invokes a custom tag for use in ColdFusion application pages. This tag processes custom tag name conflicts.

**Category**
Application framework tags

**Syntax**

```cfc
cfmodule
  attributeCollection = "collection structure"
  attribute_name1 = "valuea"
  attribute_name2 = "valueb"
  name = "tag name"
  template = "path"
  ...
</cfmodule>
```

**See also**
cfapplication, cfassociate, cflock; Creating and Using Custom CFML Tags in the Developing ColdFusion Applications

**History**
ColdFusion MX: Changed behavior when using this tag within a custom tag: if the attribute_name parameter is the same as a key element within the attributeCollection parameter, ColdFusion now uses the name value that is within the attributeCollection parameter. (Earlier releases did not process this consistently.)

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>attributeCollection</th>
<th>Optional</th>
<th>Structure. A collection of key-value pairs that represent attribute names and values. You can specify multiple key-value pairs. You can specify this attribute only once. <strong>Note:</strong> This attribute functions differently from the attributeCollection attribute that is supported by most other tags. You must specify the name and template attributes as direct cfmodule tag attributes, not in the attributeCollection structure.</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute_name</td>
<td>Optional</td>
<td>Attribute for a custom tag. You can include multiple instances of this attribute to specify the parameters of a custom tag.</td>
</tr>
<tr>
<td>name</td>
<td>Required unless template attribute is used</td>
<td>Mutually exclusive with the template attribute. A custom tag name, in the form &quot;Name.Name.Name...&quot; Identifies subdirectory, under the ColdFusion tag root directory, that contains custom tag page, for example (Windows format): &lt;cfmodule name = &quot;.Forums40. GetUserOptions&quot;&gt; This identifies the page GetUserOptions.cfm in the directory CustomTags Forums40 under the ColdFusion root directory.</td>
</tr>
<tr>
<td>template</td>
<td>Required unless name attribute is used</td>
<td>Mutually exclusive with the name attribute. A path to the page that implements the tag.</td>
</tr>
</tbody>
</table>

- Relative path: expanded from the current page.
- Absolute path: expanded by using ColdFusion mapping.
  A physical path is not valid. |

Usage

To name a ColdFusion page that contains the custom tag definition, including its path, use the template attribute. To refer to the custom tag in the ColdFusion installation directory, using dot notation to indicate its location, use the name attribute. On UNIX systems, ColdFusion searches first for a file with a name that matches the name attribute, but is all lower case. If it does not find the file, it looks for a file name that matches the attribute with identical character casing. You can use the attributeCollection attribute and explicit custom tag attributes in the same call. Within the
custom tag code, the attributes passed with attributeCollection are saved as independent attribute values, with no indication that they are grouped into a structure by the custom tag's caller. Similarly, if the custom tag uses a cfassociate tag to save its attributes, the attributes passed with attributeCollection are saved as independent attribute values, with no indication that they are grouped into a structure by the custom tag's caller. If you specify an end tag to cfmodule, ColdFusion calls your custom tag as if it had both a start and an end tag. For more information, see Handling end tags in Executing custom tags in the Developing ColdFusion Applications.

Example

```html
<h3>cfmodule Example</h3>
<p>This view-only example shows use of cfmodule to call a custom tag inline.</p>
<p>This example uses a sample custom tag that is saved in myTag.cfm in the snippets directory.
You can also save ColdFusion custom tags in the CFusionMX7\CustomTags directory.</p>

<cfset attrCollection1 = StructNew()>
<cfparam name="attrCollection1.value1" default="22">
<cfparam name="attrCollection1.value2" default="45">
<cfparam name="attrCollection1.value3" default="88">
<!--- Call the tag with CFMODULE with Name--->
<cfmodule
    Template="myTag.cfm"
    X="3"
    attributeCollection=#attrCollection1#
    Y="4">
<!--- Show the code. --->
<HR size="2" color="#0000A0">
<p>Here is one way in which to invoke the custom tag, using the TEMPLATE attribute.</p>
<cfoutput>#$HTMLCodeFormat(" <CFMODULE
    Template=""myTag.cfm"
    X=3
    attributeCollection=###attrCollection1##
    Y=4")#</cfoutput>
<p>The result: <cfoutput>#$result</cfoutput></p>
<!--- Call the tag with CFMODULE with Name.--->
</cfoutput>
<!--- Show the code. --->
<HR size="2" color="#0000A0">
<p>Here is another way to invoke the custom tag, using the NAME attribute.</p>
<cfoutput>#$HTMLCodeFormat(" <CFMODULE
    NAME='myTag'
    X=3
    attributeCollection=###attrCollection1##
    Y=4")#</cfoutput>
<p>The result: <cfoutput>#$result</cfoutput></p>
<!--- Call the tag using the shortcut notation. --->
</cfoutput>
<!--- Call the tag using the shortcut notation. --->
</cfmodule>
```

© 2014 Adobe Systems Incorporated. All rights reserved.
X="3"
attributeCollection=#attrCollection1#
Y="4">

<!--- Show the code. --->
<p>Here is the short cut to invoking the same tag.</p>
<cfoutput>#HTMLCodeFormat( "<cf_mytag
  x = 3
  attributeCollection = ##attrcollection1##
  y = 4">)#
</cfoutput>
<p>The result: <cfoutput>#result#</cfoutput></p>
cfNTauthenticate

Description

Authenticates a user name and password against the Windows NT domain on which the ColdFusion server is running, and optionally retrieves the user’s groups.

Category

Security tags

Syntax

```
<cfNTauthenticate
domain="NT domain"
password="password"
username="user name"
listGroups = "yes|no"
result="result variable"
throwOnError = "yes|no">
```

Note

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also

cflogin, cfloginuser, IsUserInAnyRole, GetAuthUser

History

ColdFusion MX 7: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>Required</td>
<td></td>
<td>Domain against which to authenticate the user. The ColdFusion J2EE server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>must be running on this domain.</td>
</tr>
<tr>
<td>password</td>
<td>Required</td>
<td></td>
<td>User’s password.</td>
</tr>
<tr>
<td>username</td>
<td>Required</td>
<td></td>
<td>User’s login name.</td>
</tr>
<tr>
<td>listGroups</td>
<td>Optional</td>
<td>No</td>
<td>Boolean value that specifies whether to include a comma-delimited list of the user's groups in the result structure.</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>result</td>
<td>Optional</td>
<td>cfntauthenticate</td>
<td>Name of the variable in which to return the results.</td>
</tr>
<tr>
<td>throwOnError</td>
<td>Optional</td>
<td>no</td>
<td>Boolean value that specifies whether to throw an exception if the validation fails. If this attribute is <em>yes</em>, ColdFusion throws an error if the username or password is invalid; the application must handle such errors in a try/catch block or ColdFusion error handler page.</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to authenticate a user against a Windows NT domain and optionally get the user’s groups. This function does not work with the Microsoft Active Directory directory service, and does nothing on UNIX and Linux systems. You typically use this tag inside a *cflogin* tag to authenticate the user for a *cfloginuser* tag, as the example shows.

⚠️ **Note**

ColdFusion must run as a user that has the privilege to authenticate other users in the specified domain.

The structure specified in the *result* attribute contains the following information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Whether the user is authenticated:</td>
</tr>
<tr>
<td></td>
<td>• yes</td>
</tr>
<tr>
<td></td>
<td>• no</td>
</tr>
<tr>
<td>groups</td>
<td>A comma-delimited list of the user’s groups in the specified domain. The structure includes this field only if the <em>listGroups</em> attribute is <em>yes</em>.</td>
</tr>
<tr>
<td>name</td>
<td>The user name; equals the tag’s <em>username</em> attribute.</td>
</tr>
</tbody>
</table>
status

The authentication status. One of the following:
- success
- UserNotInDirFailure: the user is not listed in the directory.
- AuthenticationFailure: the user is in the directory, but the password is not valid.

This tag provides two models for handling authentication: status checking and exception handling. If the throwOnError attribute is no, use the result variable’s auth and status fields to determine whether the user was authenticated and, if not, the reason for the failure. If the throwOnError attribute is yes, ColdFusion throws an exception error if the user is not valid. In this case, use try/catch error handling. The catch block must handle any authentication failure.

Example

The following example uses the auth and status fields to determine whether the user is authenticated and the failure cause. It consists of three files that you put in the same directory:

- A main cfntauthexample.cfm page that displays the name if the user is authenticated and contains a logout link.
- A login form page that is displayed if the user is not logged in.
- The Application.cfm page, which contains all the login, authentication, and logout processing code.

For a full description of login processing, see the Developing ColdFusion Applications. For information on how this example works, see the comments in the code. Save the following page as cfntauthenticatexample.cfm. To run the example, request this page in your browser or IDE.

Save the following page as loginform.cfm:

<!--- The Application.cfm page, which is processed each time a user requests this page, ensures that you log in first. --->
<cfoutput>
<h3>Welcome #GetAuthUser()#</h3>
<!--- A link to log out the user. --->
<a href="#CGI.script_name#?logout=Yes">Log Out</a>
</cfoutput>

<!--- A simple login form that posts back to the page whose request initiated the log in. --->
<h2>Please Log In</h2>
<cfform action="#CGI.script_name#"
  <!--- j_username and j_password are special names that populate cflogin tag variables. --->
  User Name: <cfinput type="text" name="j_username" value="cfqa_user1" required="Yes"></br>
  Password: <cfinput type="password" name="j_password" value="cfqa_user1" required="Yes"></br>
  Domain: <cfinput type="text" name="domain" value="rnd" required="Yes"></br>
  <input type="submit" value="Log In">
</cfform>
Save the following page as Application.cfm:

```html
<!---- If this page is executing in response to the user clicking a logout link, 
    log out the user. The cflogin tag code will then run. --->
<cfif IsDefined("URL.logout") AND URL.logout>
    <cflogout>
</cfif>

<!---- The cflogin body code runs only if a user is not logged in. --->
<cflogin>
<br>    <!---- cflogin variable exists only if login credentials are available. --->
    <cfif NOT IsDefined("cflogin")>
     <!---- Show a login form that posts back to the page whose request 
            initiated the login, and do not process the rest of this page. --->
     <cfinclude template="loginform.cfm">
    <cfabort>
    </cfelse>
<br>    <!---- Trim any leading or trailing spaces from the username and password 
          submitted by the form. --->
    <cfset theusername=trim(form.j_username)>
    <cfset thepassword=trim(form.j_password)>
    <cfset thedomain=trim(form.domain)>
    <cfntauthenticate username="#theusername#" password="#thepassword#" 
          domain="#thedomain#" result="authresult" listgroups="yes">
     <!---- authresult.auth is True if the user is authenticated. --->
    <cfif authresult.auth>
       <!---- Log user in to ColdFusion and set roles to the user's Groups. --->
       <cfloginuser name="#theusername#" password="#thepassword#" 
              roles="#authresult.groups#">
    <cfelse>
     <!---- The user was not authenticated. 
            Display an error message and the login form. --->
     <cfoutput>
    <cfif authresult.status IS "AuthenticationFailure">
       <!---- The user is valid, but not the password. --->
       <h2>The password for #theusername# is not correct<br>
            Please Try again</h2>
    </cfelse>
    <!---- There is one other status value, invalid user name. --->
    <h2>The user name #theusername# is not valid<br>
            Please Try again</h2>
    </cfif>
    <cfoutput>
    <cfinclude template="loginform.cfm">
    <cfabort>
    </cfif>
</cflogin>
```
cfoauth

The `<oauth>` tag allows you to easily integrate third-party Oauth 2 authentication provider in your application. This tag currently supports Facebook and Google authentication. Also, this tag supports Oauth providers that support the Oauth 2 protocols. For instance, Microsoft and Github.

Syntax

```xml
<cfoauth
    type = ""
    clientid = ""
    scope = ""
    state = ""
    authendpoint = ""
    secretkey = ""
    accesstokenendpoint = ""
    result = ""
    redirecturi = ""
    urlparams = ""
>
```

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type=&quot;facebook&quot;</td>
<td></td>
<td></td>
<td>Currently supported values are Facebook and Google. Implicitly supports the authentication workflow of Facebook and Google.</td>
</tr>
<tr>
<td>Type=&quot;google&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clientid</td>
<td>required</td>
<td></td>
<td>Unique ID generated while registering your application with the Oauth provider.</td>
</tr>
</tbody>
</table>
Scopes are the permissions that a developer seeks from the users. These are usually comma separated values of permissions.

Refer to the Oauth provider's documentations for more information.

For example, after Facebook authentication, if a developer wants to access an email address and then the friend lists of a user, the developer will use:

```plaintext
scope=email,read_friendlists.
```

**Note:** The scope name varies for different Oauth providers.

The state variable is used to pass back any information to your web application after the authentication and redirection are completed. Any value passed to this attribute is returned to the web application after authentication. This is useful for CSRF (Cross-site request forgery) protection. You can use ColdFusion's security-related CSRF functions for this attribute.
### authendpoint
**optional**
Either type can be specified or endpoints combination but not both.
If type is not specified, this will be used as end point URL to be invoked for user authentication.

### secretkey
**optional**
Parameter is the App Secret as displayed in your social media app’s settings.

### accesstokenendpoint
**optional**
Either type can be specified or endpoints combination but not both.
If type is not specified this will be used as end point URL to be invoked for app authentication.

### result
**optional**
A struct which will have login info of the user including login success/failure, failure reason, user name, user id.

### redirecturi
**optional**
This will default to the URL which is executing the code.
So if in oauth settings user has given app URL as :

http://domainname/appname

And the file executing the code is :

http://domainname/appname/login.cfm

The redirect URI will be :

http://domainname/appname/login.cfm

To redirect once user authentication is done.

### urlparams
**optional**
Extra options which will be passed as URL query string to authendpoint.

---

**Step 1: Register with the Oauth 2 provider**

Register your application in the Oauth 2 provider site with the relevant site-specific application details. (For instance, in Facebook you must create the Facebook App ID and App Secret).
In ColdFusion, specify your ID and App Secret in the cfoauth tag.

ColdFusion implicitly supports FB and Google. For other sites, use access token endpoint and other attributes.

The syntax is

```cfoauth
type = "Facebook|Google...."
clientid = ""
scope = ""
state = ""
authendpoint = ""
secretkey = ""
accesstokenendpoint = ""
result = ""
redirecturi = ""
urlparams = ""
>
```

redirecturi will default to the URL which is executing the code. For example, if in oauth settings user has entered the app URL as: http://domainname/appname and the file executing the code as http://domainname/appname/login.cfm, the redirect URI will be: http://domainname/appname/login.cfm.

Also, register your domain details in your social media site.

**Step 2: Understand the authentication process.**

- Once the user clicks the client site login button, the user will be prompted with a popup to grant the permissions.
- If the user is not logged into the client site, they will be prompted to login.
- If the user has already authorized your app and has already granted the permissions requested in scope, the user will be immediately redirected to your redirect_uri.

If the user has already authorized your application, but you are requesting additional permissions in the scope parameter, the user will be presented with a dialog which shows only the requested permissions that the user has not already granted your application.

**Step 3: Implementing**

The following example shows a simple authentication:
<cflogin>
  <cfoauth
type = "Facebook"
  clientid = "YOUR_CLIENT_ID"
  secretkey = "YOUR_SECRET_KEY"
  result = "res"
  redirecturi = "http://localhost:8500/doc/login.cfm" >
  <cfloginuser name = "#res.other.username#" password = "#res.access_token#" roles = "user"/>
</cflogin>

<cflocation url="http://localhost:8500/doc/index.cfm" >
cfoject

Description

Creates a ColdFusion object of a specified type.

Note

You can enable and disable this tag in the ColdFusion Administrator page, under ColdFusion Security > Sandbox Security.

Category

Extensibility tags

Syntax

The tag syntax depends on the object type. Some types use the type attribute; others do not. See the following sections:

- cfoject: .NET object
- cfoject: COM object
- cfoject: component object
- cfoject: CORBA object
- cfoject: Java or EJB object
- cfoject: web service object

Note

On UNIX, this tag does not support COM objects.

See also

cfargument, cfcomponent, cffunction, cfinvoke, cfinvokeargument, cfproperty, cfreturn: Using Java objects in the Developing ColdFusion Applications

History

ColdFusion 8:

- Added password, proxyPassword, proxyPort, proxyServer, proxyUser, refreshWSDL, userName, wsd12JavaArgs, and wsportname attributes to for use with web service objects.
- Added .NET/dotnet type and the associated assembly, port, protocol, and secure attributes.
- ColdFusion MX:
  - Changed instantiation behavior: this tag, and the CreateObject function, can now instantiate ColdFusion components (CFCs); you can use them within the cfscript tag.
  - For CORBA object: changed the Naming Service separator format for addresses from a dot to a forward slash. For example, if "context=NameService", for a class, use either of the following formats for the class parameter:
    - "/Eng/CF"
    - ".current/Eng.current/CF"
      (In earlier releases, the format was ".Eng.CF".)
  - For CORBA object: changed the locale attribute; it specifies the Java configuration that contains the properties file.
cfo bject: .NET object

Description

Creates a .NET object, that is, a ColdFusion proxy for accessing a class in a local or remote .NET assembly.

Syntax

```
<cfobject
class="class name"
name="instance name"
type=".NET|dotnet"
action="create"
assembly="absolute path"
port="6086"
protocol="tcp|http"
secure="no|yes"
server = "localhost">
```

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

.NET object, DotNetToCFType, Using Microsoft .NET Assemblies in the Developing ColdFusion Applications

History

ColdFusion 8: Added .NET and dotnet type values, and the assembly, port, protocol, and secure attributes.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Required</td>
<td></td>
<td>Name of the .NET class to instantiate as an object.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>String; reference name of the component to use in your application.</td>
</tr>
<tr>
<td>type</td>
<td>Required for .NET</td>
<td></td>
<td>Object type. Must be .NET or dotnet for .NET objects.</td>
</tr>
<tr>
<td>action</td>
<td>Optional</td>
<td>create</td>
<td>Action to take. Must be create.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Value</td>
<td>Additional Information</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------</td>
<td>------------------------</td>
</tr>
</tbody>
</table>
| assembly  | Optional    | mscorlib.dll which contains the .NET core classes. | For local .NET assemblies, the absolute path or paths to the assembly or assemblies (EXE or DLL files) from which to access the .NET class and its supporting classes. If a class in an assembly requires supporting classes that are in other assemblies, you must also specify those assemblies. You can, however, omit the supporting assemblies for the following types of supporting classes:  
  - .NET core classes (classes in mscorlib.dll)  
  - Classes in assemblies that are in the global assembly cache (GAC)  
To specify multiple assemblies, use a comma-delimited list. For remote .NET assemblies, you must specify the absolute path or paths of the local proxy JAR file or files that represent the assemblies. If you omit this attribute, and there is no local .NET installation, the tag fails without generating an error. If you omit this attribute, there is a local .NET installation, and the specified class is not in the .NET core classes, ColdFusion generates an error. |
<p>| port      | Optional    | 6086  | Port number at which the .NET-side agent is listening. |</p>
<table>
<thead>
<tr>
<th>attribute</th>
<th>description</th>
</tr>
</thead>
</table>
| protocol  | Optional: tcp. Protocol to use for communication between ColdFusion and .NET. Must be one of the following values:  
  - http: Use HTTP/SOAP communication protocol. This option is slower than tcp, but might be required for access through a firewall.  
  - tcp: Use binary TCP/IP protocol. This method is more efficient than HTTP. |
| secure    | Optional: false. Whether to secure communications with the .NET-side agent. If true, ColdFusion uses SSL to communicate with .NET. |
| server    | Optional: localhost. Host name or IP address of the server where the .NET-side agent is running. Can be in any of these forms:  
  - server name (for example, myserver)  
  - IP address (for example, 127.0.0.1)  
  You must specify this attribute to access .NET components on a remote server. |

**Usage**

The cfobject tag with a .NET or dotnet value for the type attribute creates a reference to a .NET object of a given class. Using the reference, you can access the .NET object's fields and methods. The .NET classes do not have to be local, and you can use the cfobject tag on a system that does not have .NET installed, including UNIX-based or OS-X systems.

To access .NET assemblies, do the following:

- Install the ColdFusion .NET Extension and run the .NET extension service on the system on which the assemblies are installed. You do not have to install the extension or run the extension service on a ColdFusion system that accesses only remote assemblies. For installation instructions, see *Installing and Using ColdFusion*.
- If the assemblies are located on a remote system, create Java proxies for the .NET classes that you use, copy the proxies to the ColdFusion system, and configure the remote system for access by the proxies. For
information on these steps, see Using Microsoft .NET Assemblies in the Developing ColdFusion Applications. If the .NET assemblies are on your ColdFusion system, you do not have to perform these steps.

Accessing methods and fields
You call .NET methods as you use any other ColdFusion object methods. In the simplest case, your application code uses the following format to call a .NET class method:

```
<cfobject type='.NET' name='mathInstance' class='mathClass'>
  assembly='C:/Net/Assemblies/math.dll'>
  <cfset myVar=mathInstance.multiply(1,2)>
```

If a .NET class has multiple constructors, and you do not want ColdFusion to use the default constructor to create the object, invoke a specific constructor by calling the special `init` method of the ColdFusion object with the constructor’s arguments. For example, you can use the following tags to instantiate `com.foo.MyClass(int, int)):

```
<cfobject type='.NET' class='com.foo.MyClass' assembly='c:\temp\myLib.dll' name='myObj' >
  <cfset myObj.init(10, 5)>
```

You access and change .NET class public fields by calling the following methods:

```
Get_fieldName()
Set_fieldName()
```

For example, if the .NET class has a public field named `account`, you can access and modify its value by using `Get_account()` and `Set_account()` methods, respectively.

You can access, but not modify final fields, so you can only call `Get_fieldName()` for these fields.

**Example**

The following example uses the GetProcess method of the .NET System.Diagnostics.Process class to get and display information about the processes running on the local system. Because it uses a core .NET class, for which ColdFusion automatically generates proxies, you do not have to specify an assembly name in the `cfobject` tag.

For more complex examples, including examples that use custom .NET classes, see Using Microsoft .NET Assemblies in the Developing ColdFusion Applications.
<cfobject type=".NET" name="proc" class="System.Diagnostics.Process">
<cfset processes = proc.GetProcesses()>
<cfset arrLen = arrayLen(processes)>

<table border=0 cellspacing="3" cellpadding="3">
<tr bgcolor="#33CCCC">
<td style="font-size:12px; font-weight:bold" nowrap>Process ID</td>
<td style="font-size:12px; font-weight:bold" nowrap>Name</td>
<td style="font-size:12px; font-weight:bold" nowrap>Memory (KB)</td>
<td style="font-size:12px; font-weight:bold" nowrap>Peak Memory (KB)</td>
<td style="font-size:12px; font-weight:bold" nowrap>Virtual Memory Size (KB)</td>
<td style="font-size:12px; font-weight:bold" nowrap>Start Time</td>
<td style="font-size:12px; font-weight:bold" nowrap>Total Processor Time</td>
</tr>
<cfloop from = 1 to="#arrLen#" index=i>
<cfset process = processes[i]>
<cfset id = process.Get_Id()>
<cfif id neq 0>
<cfoutput>
<tr>
<td align="right">#process.Get_Id()#</td>
<td>#process.Get_ProcessName()#</td>
<td align="right">#process.Get_PagedMemorySize()/1000#</td>
<td align="right">#process.Get_PeakPagedMemorySize()/1000#</td>
<td align="right">#process.Get_VirtualMemorySize()/1000#</td>
<td>#process.Get_StartTime()#</td>
<td>#process.Get_TotalProcessorTime()#</td>
</tr>
</cfoutput>
</cfif>
</cfloop>
</table>
cfobject: COM object

Creates and manipulates a Component Object Model (COM) object. Invokes a registered automation server object type. For information on OLEView, and about COM and DCOM, see the Microsoft OLE Development website: www.microsoft.com.

To use this tag, provide the object’s program ID or filename, the methods and properties available through the IDispatch interface, and the arguments and return types of the object’s methods. For most COM objects, you can get this information with the OLEView utility.

Note

On UNIX, the cfoobject tag does not support COM objects.

Syntax

```<cfoobject
  class = "program ID"
  name = "instance name"
  action = "create|connect"
  context = "inproc|local|remote"
  server = "server name">
  type = "com"
</cfoobject>`

Note

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also

ReleaseComObject, cfcollection, cfexecute: COM in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Required</td>
<td></td>
<td>Component ProgID for the object to invoke. When using Java stubs to connect to the COM object, the class must be the ProgID of the COM object.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>String; name for the instantiated component.</td>
</tr>
</tbody>
</table>
| action | Optional | create | • `create`: instantiates a COM object (typically, a DLL) before invoking methods or properties.  
| context | Optional | | • `connect`: connects to a COM object (typically, an EXE) running on server.  
| server | Required if `context = "Remote"` | | In Windows, if not specified, uses Registry setting.  
| type | Optional | | Object type. The value `co` specifies COM objects:  

**Example**
<h3>cfobject (COM) Example</h3>

<!--- Create a COM object as an inproc server (DLL). (class = prog-id)--->
<cfobject action = "Create"
type = "COM"
class = Allaire.DocEx1.1
name = "obj">

<!--- Call a method. Methods that expect no arguments should be called by using empty parentheses. --->
<cfset obj.Init()>

<!--- This is a collection object. It should support, at a minimum:
Property : Count
Method : Item(inarg, outarg)
and a special property called _NewEnum
--->
<cfoutput>
This object has #obj.Count# items.
<br> <HR>
</cfoutput>

<!--- Get the 3rd object in the collection. --->
<cfset emp = obj.Item(3)>
<cfoutput>
The last name in the third item is #emp.lastname#.
<br> <HR>
</cfoutput>

<!---Loop over all the objects in the collection.--->
<p>Looping through all items in the collection:
<br>
<cfloop
collection = #obj#
item = file2>
<cfoutput>Last name: #file2.lastname# <br></cfoutput>
</cfloop>
cfo
de: component object

Description

Creates an instance of a ColdFusion component (CFC) object.

Syntax

```xml
<cfobject
    component = "component name"
    name = "instance name"
    type = "component">
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

collection, ccomponent, cexecute, cfind, IsInstanceOf, creport, csearch, cfwddx; Using ColdFusion components in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>Required</td>
<td></td>
<td>Name of component to instantiate.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>String; name for the instantiated component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The name must not have a period as the first or last character.</td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
<td>component</td>
<td>The object type. You can omit this attribute or specify component. ColdFusion automatically sets the type to component.</td>
</tr>
</tbody>
</table>

Usage

When the cfo
de tag creates an instance of the CFC, ColdFusion executes any constructor code in the CFC; that is, it runs code that is not in the method definitions. On UNIX systems, ColdFusion searches first for a file with a name that matches the specified component name, but is all lowercase. If it does not find the file, it looks for a filename that matches the component name exactly, with the identical character casing.
Example

<!--- Separate instantiation and method invocation; --->
<!--- permits multiple invocations. --->
<cfobject
  name="quoteService"
  component="nasdaq.quote">
<cfinvoke
  component="#quoteService#"
  method="getLastTradePrice"
  symbol="macr"
  returnVariable="res">
    <cfoutput>#res#</cfoutput><br>
<cfinvoke
  component="#quoteService#"
  method="getLastTradePrice"
  symbol="mot"
  returnVariable="res">
    <cfoutput>#res#</cfoutput>
</cfinvoke>
**cfobject: CORBA object**

Description

Calls methods on a registered CORBA object.

Syntax

```html
<cfobject
class = "filepath or naming service"
context = "ior|nameservice"
name = "instance name"
type = "corba"
locale = "type-value arguments">
```

⚠️ Note

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

See also

cfcollection, cfexecute, cfindex, cfreport, cfsearch, cfwddx; CORBA in the *Developing ColdFusion Applications*

History

See the History section of the main `cfobject` tag page.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Required</td>
<td></td>
<td>• If <code>context = &quot;ior&quot;</code>, absolute path of file that contains string version of the Interoperable Object Reference (IOR). ColdFusion must be able to read file; it must be local to ColdFusion server or accessible on network.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If <code>context = &quot;nameservice&quot;</code>, forward slash-delimited naming context for naming service, for example: Al laire//Doc/empobj ect.</td>
</tr>
</tbody>
</table>
| context | Required | • ior: ColdFusion uses Interoperable Object Reference (IOR) to access CORBA server.  
• nameservice: ColdFusion uses naming service to access server. This option is valid only with the InitialContext of a VisiBroker Orb. |
| locale | Optional | Sets arguments for a call to `init_orb`. Use this attribute only for VisiBroker ORBs. It is available on C++, Version 3.2. The value must be in the form: 
```
{locale = "-ORBagentAddr 199.99.129.33
-ORBagentPort 19000"}
```
Each type-value pair must start with a hyphen. |
| name | Required | String; name for the instantiated component. An application uses it to reference the CORBA object's methods and attributes. |
| type | Required for CORBA | Object type. Must be `corba` for CORBA objects. |

**Usage**

ColdFusion Enterprise version 4.0 and later supports CORBA through the Dynamic Invocation Interface (DII). To use `cfobject` with CORBA objects, provide the name of the file that contains a string-formatted version of the IOR, or the object's naming context in the naming service; and the object's attributes, method names, and method signatures. User-defined types (for example, structures) are not supported.

**Example**

```
<cfobject type = "corba"
   context = "ior"
   class = "c:\myobject.ior"
   name = "GetName">
```
cfoobject: Java or EJB object

Description

Creates and manipulates a Java and Enterprise Java Bean (EJB) object.

Syntax

```xml
<cfobject
   class = "Java class"
   type = "Java"
   name = "instance name"
   action = "create">
```

Note

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also

cfeobject, cfexecute, cfindex, IsInstanceOf, cfreport, cfsearch, cfwddx; Using Java objects in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Optional</td>
<td>create</td>
<td>Only the default create action, which creates the object, is supported.</td>
</tr>
<tr>
<td>class</td>
<td>Required</td>
<td></td>
<td>The Java class.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>String; name for the instantiated component.</td>
</tr>
<tr>
<td>type</td>
<td>Required for Java</td>
<td></td>
<td>Object type. Must be java for Java and EJB objects.</td>
</tr>
</tbody>
</table>

Usage

To call Java CFXs or Java objects, ColdFusion uses a Java Virtual Machine (JVM) that is embedded in the process. You can configure JVM loading, location, and settings in the ColdFusion Administrator. Any Java class available in the class path that is specified in the ColdFusion Administrator can be loaded and used from ColdFusion, by using the cfoobject tag.

Access Java methods and fields
1. Call the `cfobject` tag, to load the class. See the example code.
2. Use the `init` method with appropriate arguments, to call a constructor. For example:

   ```cfml
   <cfset ret = myObj.init(arg1, arg2)>
   ```

Calling a public method on the object without first calling the `init` method results in an implicit call to the default constructor. Arguments and return values can be any Java type (simple, array, object). ColdFusion makes the conversions if strings are passed as arguments, but not if they are received as return values. Overloaded methods are supported if the number of arguments is different.

**Calling EJBs**

To create and call EJB objects, use the `cfobject` tag. In the second example in the following section, the WebLogic JNDI is used to register and find EJBHome instances.

**Example**

```cfml
<!---- Example of a Java Object, this cfobject call loads the class MyClass but does not create an instance object. Static methods and fields are accessible after a call to cfobject. ---->
<cfobject
  action = "create"
  type = "java"
  class = "myclass"
  name = "myobj">

<!---- Example of an EJB - The cfobject tag creates the Weblogic Environment object, which is used to get InitialContext. The context object is used to look up the EJBJndiHome interface. The call to Create() results in getting an instance of stateless session EJB. ---->
<cfobject
  action = "create"
  type = "java"
  class = "weblogic/jndi/Environment"
  name = "wlEnv">
  <cfset ctx = wlEnv.getInitialContext()>
  <cfset ejbHome = ctx.lookup("statelessSession.TraderHome")>
  <cfset trader = ejbHome.Create()>
  <cfset value = trader.shareValue(20, 55.45)>
  <cfoutput>
    Share value = #value#
  </cfoutput>
  <cfset value = trader.remove()>
```
**cfojbeject: web service object**

**Description**

Creates a web service proxy object.

**Syntax**

```xml
<cfojbeject
    name = "local name">
    webservice= "service identifier"
    password = "string"
    proxyPassword = "string"
    proxyPort = "port number"
    proxyServer = "URL or IP address"
    proxyUser = "string"
    refreshWSDL = "no|yes"
    type = "webservice"
    username = "string"
    wsdl2javaArgs = "argument string"
    wsportname = "port name">
</cfojbeject>
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfcollection, cfexecute, cfindex, cfreport, cfsearch, cfwddx; Consuming web services in Using Web Services in the Developing ColdFusion Applications

**History**

See the History section of the main cfojbeject tag page.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Local name for the web service. String.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Required</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>webservice</td>
<td>Required</td>
<td>One of the following:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The absolute URL of the web service.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The name (string) assigned in the ColdFusion Administrator to the web service.</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td>Password set in the Administrator, if any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The password to use to access the web service. If the <code>webservice</code> attribute specifies a web service name configured in the ColdFusion Administrator, overrides any password specified in the Administrator entry.</td>
<td></td>
</tr>
<tr>
<td>proxyPassword</td>
<td>Optional</td>
<td><code>http.proxyPassword</code> system property, if any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The user's password on the proxy server.</td>
<td></td>
</tr>
<tr>
<td>proxyPort</td>
<td>Optional</td>
<td><code>http.proxyPort</code> system property, if any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The port to use on the proxy server.</td>
<td></td>
</tr>
<tr>
<td>proxyServer</td>
<td>Optional</td>
<td><code>http.proxyHost</code> system property, if any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The proxy server required to access the web service URL.</td>
<td></td>
</tr>
<tr>
<td>proxyUser</td>
<td>Optional</td>
<td><code>http.proxyUser</code> system property, if any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The user ID to send to the proxy server.</td>
<td></td>
</tr>
<tr>
<td>refreshWSDL</td>
<td>Optional</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>yes</code>: reloads the WSDL file and regenerates the artifacts used to consume the web service</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>no</code></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
<td>The object type. You can omit this attribute or specify <code>webservice</code>.</td>
<td></td>
</tr>
<tr>
<td><strong>username</strong></td>
<td>Optional</td>
<td>User name set in the Administrator, if any</td>
<td>The user name to use to access the web service. If the <code>webservice</code> attribute specifies a web service configured name in the ColdFusion Administrator, overrides any user name specified in the Administrator entry.</td>
</tr>
<tr>
<td><strong>wsdl2javaArgs</strong></td>
<td>Optional</td>
<td>A string that contains a space-delimited list of arguments to pass to the WSDL2Java tool that generates Java stubs for the web services. Useful arguments include the following:</td>
<td></td>
</tr>
<tr>
<td><strong>wsportname</strong></td>
<td>Optional</td>
<td>First port in the WSDL</td>
<td>The port name for the web service. This value is case sensitive and corresponds to the <code>port</code> element's <code>name</code> attribute under the <code>service</code> element. Specify this parameter if the web service contains multiple ports.</td>
</tr>
</tbody>
</table>

Usage
Instantiates a proxy object for a web service. You can enter the absolute URL in this tag, or refer to a web service that is entered in the ColdFusion Administrator. To minimize potential code maintenance, enter the web service in the Administrator, and then refer to that name in this tag.
**cfobjectcache**

**Description**

Flushes the query cache.

**Category**

Database manipulation tags

**Syntax**

```xml
<cfobjectcache
    action = "clear">
```

**Note**

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

**See also**

cfobject

**History**

ColdFusion 5: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>clear: clears queries from the cache in the Application scope.</td>
</tr>
</tbody>
</table>
**cfoutput**

**Description**

Displays output that can contain the results of processing ColdFusion variables and functions. Can loop over the results of a database query.

**Category**

Data output tags

**Syntax**

```plaintext
<cfoutput
group = "query column"
groupCaseSensitive = "yes|no"
maxRows = "maximum rows to display"
query = "query name"
startRow = "start row">
</cfoutput>
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfcol, cfcontent, cfdirectory, cftable

**History**

ColdFusion 4.5.0: Added the groupCaseSensitive attribute.

**Attributes**

| Attribute          | Req/Opt | Default | Description |
|--------------------|---------|---------|-------------|-------------|
|                    |         |         |             |             |
group | Optional | Query column to use to group sets of records. Eliminates adjacent duplicate rows when data is sorted. Use if you retrieved a record set ordered on one or more a query columns. For example, if a record set is ordered on "Customer_ID" in the cfquery tag, you can group the output on "Customer_ID."

groupCaseSensitive | Optional | yes | Boolean. Whether to consider the case in grouping rows.

maxRows | Optional | Displays all rows | Maximum number of rows to display.

query | Optional | Name of cfquery from which to draw data for output section.

startRow | Optional | 1 | Row from which to start output.

Usage

In the cfoutput tag body, ColdFusion treats text that is surrounded by number signs (#) as a ColdFusion variable or function call. For example, the following code displays the text "Hello World!":

```cfc
<cfset myVar="Hello World!">
<cfoutput>#myVar#</cfoutput>
```

When you specify a query attribute, this tag loops over the query rows and produces output for each row within the range specified by the startRow and maxRows values, and groups or eliminates duplicate entries as specified by the grouping attribute values, if any. It also sets the query.currentRow variable to the current row being processed. If you nest cfoutput blocks that process a query, you specify the query and group attributes at the top-most level; you can specify a group attribute for each inner block except the innermost cfoutput block. This tag requires an end tag.

Example

```cfc
<!--- EXAMPLE: This example shows how cfoutput operates. --->
<!--- Run a sample query. --->
<cfquery name = "GetCourses" dataSource = "cfdocexamples">
```
SELECT Dept_ID, CorName, CorLevel
FROM courseList
ORDER by Dept_ID, CorLevel, CorName
</cfquery>

<h3>cfoutput Example</h3>

<p>cfoutput tells ColdFusion Server to begin processing, and then to hand back control of page rendering to the web server.
</p><p>For example, to show today's date, you could write #DateFormat("#Now()#"). If you enclosed that expression in cfoutput, the result would be#DateFormat(Now())#</cfoutput>.</p>

<p>In addition, cfoutput may be used to show the results of a query operation, or only a partial result, as shown:</p>

<p>There are #getCourses.recordCount# total records in our query. Using the maxRows parameter, we are limiting our display to 4 rows.
</p><p>#Dept_ID# #CorName# #CorLevel##<br></cfoutput></p>

<p>EXAMPLE: The next example uses the group attribute to eliminate duplicate lines from a list of course levels taught in each department.</p>

<p><cfquery name = "GetCourses" dataSource = "cfdocexamples" ></p>

SELECT Dept_ID, CorLevel
FROM courseList
ORDER by Dept_ID, CorLevel</cfquery>

<p>There are #Dept_ID# #CorName# #CorLevel#<br></cfoutput></p>

<p>cfoutput can also show the results of a more complex expression, such as getting the day of the week from today's date. We first extract the integer representing the Day of the Week from the server function Now() and then apply the result to the DayofWeekAsString function:</p>

<br>Today is #DayofWeekAsString(DayofWeek(Now()))#
<br>Today is <cfoutput>#DayofWeekAsString(DayofWeek(Now()))#</cfoutput></p>

<p>EXAMPLE: This last example shows nested cfoutput tags:</p>

<cfquery datasource="cfdocexamples" name="empSalary">
SELECT Emp_ID, firstname, lastname, e.dept_id, salary, d.dept_name
FROM employee e, departmt d
WHERE e.dept_id = d.dept_id
ORDER BY d.dept_name
</cfquery>

<!---- Outer cfoutput. ---->
<cfoutput query="empSalary" group="dept_id">
<h2>#dept_name##</h2>
<table width="95%" border="2" cellspacing="2" cellpadding="2">
<tr>
<th>Employee</th>
<th>Salary</th>
</tr>
<tr>
<th>Employee</th>
<th>Salary</th>
</tr>
</table>
</cfoutput>

<!---- Inner cfoutput. ---->
<cfoutput>
<tr><td>#{empSalary.lastname#, empSalary.firstname#}</td><td align="right">#{DollarFormat(empSalary.salary)#}</td></tr>
<cfset deptTotal = deptTotal + empSalary.salary>
</cfoutput>
<tr><td align="right">Total</td><td align="right">#{DollarFormat(deptTotal)#}</td></tr>
<cfset deptTotal = 0>
</table>
</cfoutput>

#back to top
Tags p-q

This section lists tags starting with alphabets P-Q with their description.

- cfparam
- cfpdf
- cfpdfform
- cfpdfformparam
- cfpdfparam
- cfpdfsubform
- cfpod
- cfpop
- cfpresentation
- cfpresentationslide
- cfpresenter
- cfprint
- cfprocessingdirective
- cfpromparam
- cfprocresult
- cfprogressbar
- cfproperty
- cfquery
- cfqueryparam
**cfparam**

**Description**

Tests for the existence of a parameter (that is, a variable), validates its data, and, if a default value is not assigned, optionally provides one.

**History**

ColdFusion 10: Added the attribute {{maxLength}}

ColdFusion MX 7:
- Added min, max, and pattern attributes.
- Added creditcard, email, eurodate, float, integer, range, regex, regular_expression, ssn, social_security_number, time, URL, USdate, XML, and zipcode values of the type attribute.

**Category**

Variable manipulation tags

**Syntax**

```
<cfparam
  name = "parameter name"
  default = "value"
  max = "value"
  maxLength = "number"
  min = "value"
  pattern = "regular expression"
  type = "data_type">
```

**See also**

cfcookie, cfregistry, cfsavecontent, cfschedule, cfset; Validating data with the IsValid function and the cfparam tag in the Developing ColdFusion Applications

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Name of the parameter (variable) to test (such as &quot;Client.Email&quot; or &quot;Cookie.BackgroundColor&quot;). If omitted, and if the parameter does not exist, an error is thrown.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>Optional</td>
<td>Value to set parameter to if it does not exist. Any expression used for the default attribute is evaluated, even if the parameter exists. The result is not assigned if the parameter exists, but if the expression has side effects, they still occur.</td>
<td></td>
</tr>
<tr>
<td>max</td>
<td>Optional</td>
<td>The maximum valid value; used only for range validation.</td>
<td></td>
</tr>
<tr>
<td>maxLength</td>
<td>Optional</td>
<td>Used to specify the maximum character length of email, url, and string.</td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>Optional</td>
<td>The minimum valid value; used only for range validation.</td>
<td></td>
</tr>
<tr>
<td>pattern</td>
<td>Optional</td>
<td>A JavaScript regular expression that the parameter must match; used only for regex or regular_expression validation.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The valid format for the data; one of the following. For detailed information on validation algorithms, see Validating form data using hidden fields in Validating data in the Developing ColdFusion Applications.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- any: any type of value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- array: an array of values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- binary: a binary value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- boolean: a Boolean value: yes, no, true, false, or a number.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- creditcard: a 13-16 digit number conforming to the mod10 algorithm.</td>
<td></td>
</tr>
</tbody>
</table>
- **date or time**: a date-time value.
- **email**: a valid e-mail address.
- **eurodate**: a date-time value. Any date part must be in the format dd/mm/yy. The format can use /, -, or . characters as delimiters.
- **float or numeric**: a numeric value.
- **guid**: a Universally Unique Identifier of the form "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" where x is a hexadecimal number.
- **integer**: an integer.
- **query**: a query object.
- **range**: a numeric range, specified by the min and max attributes.
- **regex or regular_expression**: matches input against pattern attribute.
- **ssn or social_security_number**: a U.S. social security number.
- **string**: a string value or single character.
- **struct**: a structure.
- **telephone**: a standard U.S. telephone number.
- **URL**: an http, https, ftp, file, mailto, or news URL.
- **UUID**: a ColdFusion Universally Unique Identifier, formatted XXX XXXXX-XXXX-XXXX-X XXXXXXXXXX, where x is a hexadecimal number. See [CreateUUID](#).
- **USdate**: a U.S. date of the format mm/dd/yy, with 1-2 digit days and months, 1-4 digit years.
variableName: a string formatted according to ColdFusion variable naming conventions.

xml: XML objects and XML strings.

zipcode: U.S., 5- or 9-digit format ZIP codes.

Usage

You can use this tag to make the following tests:

- To test whether a required variable exists, use this tag with only the name attribute. If it does not exist, ColdFusion stops processing the page and returns an error.
- To test whether a required variable exists, and that it is of the specified type, use this tag with the name and type attributes. If the variable does not exist or its value is not of the specified type, ColdFusion returns an error.
- To set a default value for the variable, use this tag with the name and default attributes. If the variable does not exist, it is created and set to the default attribute value. If the variable exists, processing continues; the value is not changed.

If you specify variableName for the type attribute, the parameter's value must be a string that is in ColdFusion variable name format; that is, starts with a letter, underscore (_), or Unicode currency symbol, and contains letters, numbers, underscores, periods, and Unicode currency symbols, only. ColdFusion does not check whether the parameter value corresponds to an existing ColdFusion variable.

To improve performance, avoid using the cfparam tag in ColdFusion functions, including in CFC methods. Instead, place the cfparam tags in the body of the CFML pages.

Example
<!--- This example shows how to use CFPARAM to define default values for page variables. --->
<cfparam name = "storeTempVar" default = "my default value">
<cfparam name = "tempVar" default = "my default value">

<!--- Check if form.tempVar was passed. --->
<cfif IsDefined("form.tempVar") is "True">
   <!--- Check if form.tempVar is not blank. --->
   <cfif form.tempVar is not ">">
      <!--- If not, set tempVar to value of form.tempVar --->
      <cfset tempVar = form.tempVar>
   </cfif>
</cfif>

<body>
<h3>cfparam Example</h3>
<p>cfparam is used to set default values so that a developer does not have to check for the existence of a variable using a function like IsDefined.</p>
<p>The default value of our tempVar is "<cfoutput>#StoreTempVar#</cfoutput>"</p>

<!--- Check if tempVar is still the same as StoreTempVar and that tempVar is not blank. --->
<cfif tempVar is not #StoreTempVar# and tempVar is not "">
   <h3>The value of tempVar has changed: the new value is</h3>
   <cfoutput>#tempVar#</cfoutput></h3>
</cfif>

<p>
<form action = "cfparam.cfm" method = "post">
   Type in a new value for tempVar, and hit submit:<br>
   <input type = "Text" name = "tempVar">
   <input type = "Submit" name = "" value = "submit">
</form>
</p>
cftp

Description

Manipulates existing PDF documents. The following list describes some of the tasks you can perform with the cfpdf tag:

- Merge several PDF documents into one PDF document.
- Delete pages from a PDF document.
- Merge pages from one or more PDF documents and generate a new PDF document.
- Linearize PDF documents for faster web display.
- Remove interactivity from forms created in Acrobat to generate flat PDF documents.
- Encrypt and add password protection to PDF documents.
- Generate thumbnail images from PDF documents or pages.
- Add or remove watermarks from PDF documents or pages.
- Retrieve information associated with a PDF document, such as the software used to generate the file or the author, and set information for a PDF document, such as the title, author and keywords.
- Create PDF portfolios
- Add and remove header/footer from PDF documents
- Optimize PDF documents

History

ColdFusion 11 has added the following new attributes to the <cfpdf> tag:

- keystore
- keystorepassword
- keyalias
- keypassword
- author
- signaturefieldName
- unsignAll
- height
- width

ColdFusion 8: Added this tag. ColdFusion 9: Added new attributes: jpgdpi, maxBreadth, noAttachments, leftMargin, algo, noMetadata, noBookMarks, noJavaScripts, useStructure, noFonts, text, noComments, encodeAll, numberFormat, compressTIFFs, addQuads, rightMargin, topMargin, bottomMargin, noThumbnails, align, noLinks, maxLength, hires, hScale, overridepage, honourspaces, maxScale, packagge, vScale

Category

Data output tags

Syntax

Add a watermark to a PDF document
<cfpdf
  required
action = "addwatermark"
source = "absolute or relative pathname to a PDF file|PDF document variable|
cfdocument variable"

one of the following:
copyfrom = "absolute or relative pathname to a PDF file from which the first page is used as a watermark"
image = "absolute or relative pathname to image file|image variable used as a watermark"

optional
foreground = "yes|no"
isBase64 = "yes|no"
opacity = "watermark opacity"
overwrite = "yes|no"
pages = "page or pages to add the watermark"
password = "user or owner password for the PDF source file"
position = "position on the page where the watermark is placed"
rotation = "degree of rotation of the watermark"
showonprint = "yes|no">

one of the following:
destination = "PDF output file pathname"
name = "PDF document variable name"
image = "image file name to be used as the footer"
text = "text to be used in the footer"

add headers
<cfpdf
required
action = "addheader"
source = "absolute or relative pathname to a PDF file|PDF document variable|
cfdocument variable"
pages = "page or pages to add the footer"
optional
isBase64 = "yes|no"
overwrite = "yes|no"
password = "user or owner password for the PDF source file"
showonprint = "yes|no">

align = "left|right|center"
leftmargin = "value of the header left marign"
rightmargin = "value of the header right margin"
numberformat = "LOWERCASEROMAN|NUMERIC|UPPERCASEROMAN" "!"----used with either _PAGENUMBER or _LASTPAGENUMBER-->
opacity = "header opacity"
topmargin = "value of the top margin of the header"

one of the following:
destination = "PDF output file pathname"
name = "PDF document variable name"
text = _PAGELABEL: add current page label|_LASTPAGELABEL: add last page label|
_PAGENUMBER: add current page number|_LASTPAGENUMBER: add last page number \
text for the header. You can also add a normal text string.
image = "image file name to be used as the header"

add footer
<cfpdf
required
action = "addfooter"
source = "absolute or relative pathname to a PDF file|PDF document variable|
cfdocument variable"
pages = "page or pages to add the footer"
optional
isBase64 = "yes|no"
overwrite = "yes|no"

password = "user or owner password for the PDF source file"
showonprint = "yes|no">
destination = "PDF output file pathname"
name = "PDF document variable name"
align = "left|right|center"
one of the following:
image = "image file name to be used as the footer"
text =  _PAGELABEL: add current page label|_LASTPAGELABEL: add last page label|_PAGENUMBER: add current page number|_LASTPAGENUMBER: add last page number \
text for the header
leftmargin = "value of the footer left margin"
rightmargin = "value of the footer right margin"
numberformat
opacity = "footer opacity"
bottommargin = "value of the bottom margin"

Delete pages from a PDF document
<cfpdf
required
action = "deletepages"
pages = "page or pages to delete"
source = "absolute or relative pathname to a PDF file|PDF document variable|
cfdocument variable"
optional
overwrite = "yes|no"
password = "PDF source file password"
one of the following:
destination = "PDF output file pathname"
name = "PDF document variable name">

Delete headers and footers
<cfpdf
required
action = "removeheaderfooter"
source = "absolute or relative pathname to a PDF file|PDF document variable|
cfdocument variable"
optional
overwrite = "yes|no"
pages = "page or pages to add the watermark"
password = "user or owner password for the PDF source file"
one of the following:
destination = "PDF output file pathname"
name = "PDF document variable name"

Retrieve information about a PDF document
<cfpdf
required
action = "getinfo"
name = "structure variable name"
source = "absolute or relative pathname to a PDF file|PDF document variable|
cfdocument variable"
optional
password = "PDF source file password">

Merge PDF documents into an output PDF file
<cfpdf
required
action = "merge"
one of the following:
directory = "directory of PDF files to merge"
source = "comma-separated list of PDF source files|absolute or relative pathname
to a PDF file|PDF document variable|cfdocument variable"
<cfpdfparam>
    \\required only when package is specified as true
    order = "name|time"
    one of the following if <cfpdfparam> is specified:
    name = "PDF document variable name"
    destination = "PDF output file pathname"
optional
    package = "true|false" <!---create PDF packages if set to true. You can provide
        description in cfpdfparam tag, such as <cfpdfparam file="filename desc=""/>--->
    ascending = "yes|no"
    keepBookmark = "yes|no"
    overwrite = "yes|no"
    pages = "pages to merge in PDF source file"
    password = "PDF source file password"
    stopOnError = "yes|no"
    \\one of the following:
    destination = "PDF output file pathname"
    name = "PDF document variable name">
Use DDX instructions to manipulate PDF documents
<cfpdf
    required
    ddxfile = "DDX filepath|DDX string"
    inputfiles = "#inputStruct#"
    outputfiles = "#outputStruct#"
    name = "structure name">
optional
    action="processddx"
Set passwords and encrypt PDF documents
<cfpdf
    required
    action = "protect"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
        cfdocument variable"
    at least one of the following:
    newUserPassword = "password"
    newOwnerPassword = "password"
    if newOwnerPassword is specified:
        permissions = "All|AllowAssembly|AllowDegradedPrinting|AllowCopy|AllowFillIn|AllowModifyAnnotation
                       s|AllowModifyContents|AllowPrinting|AllowScreenReaders|AllowSecure|None|
        comma-separated list"
optional
    destination = "PDF output file pathname"
    encrypt = "RC4_40|RC4_128|RC4_128M|AES_128|none"
    overwrite = "yes|no"
    password = "source file password">
Name a PDF document variable
<cfpdf
    required
    action = "read"
    name = "PDF document variable name"
    source = "absolute or relative pathname to a PDF file|PDF document variable|
Remove a watermark from a PDF document
<cfpdf
  required
  action = "removeWatermark"
  source = "absolute or relative pathname to a PDF file|PDF document variable|cfdocument variable"
  optional
  overwrite = "yes|no"
  pages = "page or pages from which to remove the watermark"
  password = "PDF source file password">
  one of the following:
  destination = "PDF output file pathname"
  name = "PDF document variable name"
</cfpdf>

Set information about a PDF document
<cfpdf
  required
  action = "setinfo"
  info = "#structure variable name#"
  source = "absolute or relative pathname to a PDF file|PDF document variable|cfdocument variable"
  optional
  destination = "PDF output file pathname"
  overwrite = "yes|no"
  password = "PDF source file password">
</cfpdf>

Generate thumbnails from pages in a PDF document
<cfpdf
  required
  action = "thumbnail"
  source = "absolute or relative pathname to a PDF file|PDF document variable|cfdocument variable"
  optional
  destination = "directory path where the thumbnail images are written"
  format = "png|jpeg|tiff"
  imagePrefix = "string used as a prefix in the output filename"
  overwrite = "yes|no"
  password = "PDF source file password">
  pages = "page or pages to make into thumbnails"
  resolution = "low|high"
  scale = "percentage between 1 and 100"
  transparent = "yes|no">
  hires = "yes|no"
  overridepage = "yes|no"
  compressstiffs = "yes|no"
  maxscale = "maximum scale of the thumbnail"
  maxlength = "maximum length of the thumbnail"
  maxbreadth = "maximum width of the thumbnail"
</cfpdf>

Write a PDF document to an output file
<cfpdf
  required
  action = "write"
  source = "absolute or relative pathname to a PDF file|PDF document variable|cfdocument variable"
one of the following
destination = "PDF output file pathname"
name = #PDF variable# <!---new variable support added now-->
optional
flatten = "yes|no"
overwrite = "yes|no"
password = "PDF source file password"
saveOption = "linear|incremental|full"
version = "1.1|1.2|1.3|1.4|1.5|1.6">
encodeall = "yes|no"

Reduce the quality of a PDF document
<cfpdf
required
action = "optimize"
source = "absolute or relative path of the PDF file|PDF document variable|
cfdocument variable"
algo = "bilinear|bicubic|nearest_neighbour" <!---algorithm for image
downsampling-->
pages = "*" <!----page numbers associated with the objects in the PDF document-->
optional
vscale= "Vertical scale of the image to be modified. Valid values are vscale>0"
hscale="Horizontal scale of the image to be modified. Valid values are hscale<1"
destination = "PDF output file pathname"
name = "PDF document variable"
noattachments = "Discard all attachments"
nobookmarks = "Discard all bookmarks"
nocomments = "Discard all comments"
nofonts = "Discard all fonts"
nojavascripts = "Discard all JavaScript actions"
nolinks = "Discard external cross-references"
nometadata = "Discard document information and metadata"
nothumbnails = "Discard embedded page thumbnails"
overwrite = "true" <!---Overwrite the specified object in the PDF document--->
password = "" <!--- PDF document password--->

Extract text
<cfpdf
required
action="extracttext" <!----extract all the words in the PDF.---->
source= "absolute or relative path of the PDF file|PDF document variable|
cfdocument variable"
pages = "*" <!----page numbers from where the text needs to be extracted from the
PDF document-->
optional
addquads = "add the position or quadrants for the text in the PDF"
honourspaces = "true|false"
overwrite = "true" <!---Overwrite the specified object in the PDF document-->
password = "" <!--- PDF document password--->
type = "string|xml" <!---format in which the text needs to be extracted--->
one of the following:
destination = "PDF output file pathname"
name = "PDF document variable"
usestructure = "true|false"

Extract image
<cfpdf
required
action = "extractimage" <!----extract images and save it to a directory--->

source = "absolute or relative path of the PDF file|PDF document variable|
    cfdocument variable"
pages = "*" <!---page numbers from where the images need to be extracted-->
optional
overwrite = "true|false" <!---overwrite any existing image when set to true-->
format = "png|tiff|jpg" <!---format in which the images should be extracted-->
imageprefix = "*" <!---the string that you want to prefix with the image name-->
password = "" <!--- PDF document password-->
destination = "PDF output file pathname"

Page level transformations
<cfpdf
    required
    action = "transform"
    source = "absolute or relative path of the PDF file|PDF document variable|
        cfdocument variable"
    pages = "page or pages to be transformed"
    optional
    hscale = "value of the horizontal scale of the page"
    overwrite = "yes|no"
    password = "PDF source file password"
    position = "x, y" <!---value in pixels-->
    rotation = "0|90|180|270"
    vscale = "length of the page to be transformed"
    one of the following:
destination = "Path of the directory where the PDF document will be saved"
name = "PDF document variable"

⚠️ **Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfdocument, cfdocumentsection, cfpdfform, cfpdfformparam, cfpdfparam, cfpdfsubform, cfprint, IsDDX, IsPDFFile, IsPDFObject, Assembling PDF Documents in the Developing ColdFusion Applications

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>N/A</td>
<td>Optional</td>
<td>Action to take:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>addWatermark</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>deletePages</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>getInfo</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>merge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>processddx</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>protect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>read</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>removeWatermark</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>setInfo</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sign</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>thumbnail</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>write</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>optimize</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>extracttext</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>extractimage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>addheader</td>
</tr>
</tbody>
</table>
addfooter
removeheaderfooter
transform
unsign
validatesignature

**New in ColdFusion 11**—Validates all the signatures in the document. The name attribute will then store the resultant struct of this operation. The name attribute will contain 2 elements:

- A Boolean field that indicates whether all the signatures have been deemed valid or not.
- An array list consisting of the names of all the signatures that were invalid. If all the signatures are valid, the array list will be empty.

readsatisfiedfields
### New in ColdFusion 11

Reads all the signature fields and returns a query object in the variable indicated by the attribute name. This query object will have 4 columns:

- **Name**: A fully qualified name of the signature field
- **Certifiable**: Indicates if this field can contain an author signature or not
- **Signable**: Indicates whether this field can be signed or not
- **Is signed**: Indicates whether this field is already signed or not

<table>
<thead>
<tr>
<th>addquads</th>
<th>extracttext</th>
<th>Optional</th>
<th>Add the position or quadrants of the thumbnail</th>
</tr>
</thead>
<tbody>
<tr>
<td>align</td>
<td>addheader</td>
<td>addfooter</td>
<td>Optional</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Requirements</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>--------------</td>
<td>-------</td>
</tr>
<tr>
<td><code>algo</code></td>
<td>Specifies the algorithm for image downsampling. The values are bilinear, bicubic, and nearest_neighbour</td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td><code>ascending</code></td>
<td>Order in which the PDF files are sorted: yes: Files are sorted in ascending order no: Files are sorted in descending order Applicable only when you specify the <code>directory</code> attribute.</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td><code>author</code></td>
<td>Business transactions, including financial, legal, and other regulated transactions, require high assurance when signing documents. When documents are distributed electronically, it is important that recipients can: Verify document authenticity – confirm the identity of person who signed the document Verify document integrity – confirm that the document has not been altered in transit Author-based signatures provide both of these security services. If attribute is not mentioned then it is treated as <code>author=false</code></td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td><code>bottomMargin</code></td>
<td>Specifies the value of the <code>bottomMargin</code></td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>copyFrom</td>
<td>addWatermark</td>
<td>Optional</td>
<td>Pathname of the PDF document from which to use the first page as a watermark</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>compressstiffs</td>
<td>thumbnail</td>
<td>Optional</td>
<td>Compress thumbnail which are in TIFF format.</td>
</tr>
<tr>
<td>ddxfile</td>
<td>processddx</td>
<td>Required</td>
<td>Pathname of the DDX file, or a string with DDX instructions</td>
</tr>
<tr>
<td>destination</td>
<td></td>
<td></td>
<td>Required for the write actionOptional for all other actions</td>
</tr>
<tr>
<td></td>
<td>addWatermark</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>archive</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>deletePages</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>merge</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>protect</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>removeWatermark</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sign</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>setInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>thumbnail</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>write</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>optimize</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>extracttext</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>extractimage</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>addheader</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>addfooter</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>removeheaderfooter</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>transform</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pathname of the modified PDF document. If the destination file exists, set the overwrite attribute to yes. If the destination file does not exist, ColdFusion creates the file, if the parent directory exists.
You can specify the destination attribute or the name attribute, but not both.
For the thumbnail action, the destination is the directory path where the images are written. If you specify a relative pathname to the destination directory, the destination directory is relative to the template directory. If you do not specify a destination directory, ColdFusion creates a directory called thumbnails in the directory in the template directory. For the optimize action, destination is the path where the PDF document which needs to be optimized is located. For extracttext and extractimage, destination is the path of the PDF document from which the text or image needs to be extracted.
For addheader, addfooter, removeheader footer, destination is the path of the PDF document where you need to add a header or footer, or remove the header and footer.
For transform, destination specifies the directory path of the PDF document where you need to perform page level transformations.
<table>
<thead>
<tr>
<th><strong>directory</strong></th>
<th><strong>merge</strong></th>
<th><strong>Optional</strong></th>
<th>Directory of the PDF documents to merge. Specify either the <code>directory</code> attribute or the <code>source</code> attribute. If you specify the <code>directory</code> attribute, ColdFusion orders the documents by filename in descending order, by default. To change the order of the files, use the <code>order</code> attribute.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>encodeall</strong></td>
<td><strong>write</strong></td>
<td><strong>Optional</strong></td>
<td>Encode streams that are not encoded to optimize page content</td>
</tr>
<tr>
<td><strong>encrypt</strong></td>
<td><strong>protect</strong></td>
<td><strong>Optional</strong></td>
<td>Encryption type for the PDF output file: <code>RC4_40</code>, <code>RC4_128</code>, <code>RC4_128M</code>, <code>AES_128</code>, <code>None</code>. For more information, see the section <em>Encryption for PDF documents</em>.</td>
</tr>
<tr>
<td><strong>flatten</strong></td>
<td><strong>write</strong></td>
<td><strong>Optional</strong></td>
<td>Applies to forms created in Acrobat only (not forms created in LiveCycle); specifies whether interactivity is turned off: <code>yes</code>: the form fields are no longer interactive. <code>no</code>: the form fields remain interactive.</td>
</tr>
</tbody>
</table>
| **foreground** | **addWatermark** | Optional | Placement of the watermark on the page:

yes: the watermark appears in the foreground (over the page content).

no: the watermark appears in the background (behind the page content).

| **format** | **thumbnail** | Optional | File type of thumbnail image output:

jpg
tiffpng

| **hires** | **thumbnail** | optional | Sets a high resolution for the thumbnail if set to yes.

| **height** | **sign** | Optional | Height of the signature field.

| **honourspaces** | **extracttext** | optional | Set this option to "true", for improved readability and spacing.

| **hscale** | **optimize** | optional | Horizontal scale of the image to be modified. Valid values are hscale<1.

| **image** | **addWatermark** | Optional | Image used as a watermark. You can specify a pathname, a variable that contains an image file, or a ColdFusion image variable. |
| **imagePrefix** | **thumbnail** | **Optional** | Prefix used for each image thumbnail file generated. The image filenames use the format: `imagePrefixpagen.format`. For example, the thumbnail for page 1 of a document with the `imagePrefix` attribute set to `myThumbnail` is `myThumbnail_page_1.jpg`.

| **info** | **setInfo** | **Required** | Structure variable for relevant information, for example, `#infoStruct`. You can specify the Author, Subject, Title, and Keywords for the PDF output file.

| **inputFiles** | **processddx** | **Required** | Structure that maps the PDF source files to the input variables in the DDX file, or a string of elements and their pathname.

| **isBase64** | **addWatermark** | **Optional** | Valid only when the `image` attribute is specified. Specifies whether the image used as a watermark is in Base64 format:

- `yes`: the image is in Base64 format.
- `no`: the image is not in Base64 format.

| **keepBookmark** | **merge** | **Optional** | Specifies whether bookmarks from the source PDF documents are retained in the merged document:

- `yes`: the bookmarks are retained.
- `no`: the bookmarks are removed.
<table>
<thead>
<tr>
<th>Key</th>
<th>Sign</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyalias</td>
<td>sign</td>
<td>Optional</td>
<td>Alias of the key with which the certificate and private key are stored in the keystore. If it is not specified, the first entry in the keystore is chosen as the alias.</td>
</tr>
<tr>
<td>keystore</td>
<td>sign</td>
<td>Required</td>
<td>The location of the keystore file. For example, C:\OpenSSL\bin\keystore.jks.</td>
</tr>
<tr>
<td>keypassword</td>
<td>sign</td>
<td>Optional</td>
<td>Password for your private key. If not specified the keystorepassword is used.</td>
</tr>
<tr>
<td>keystorepassword</td>
<td>sign</td>
<td>Required</td>
<td>The password of the keystore.</td>
</tr>
<tr>
<td>leftmargin</td>
<td>addheader</td>
<td>Optional</td>
<td>Specifies the value of the header left margin</td>
</tr>
<tr>
<td>maxbreadth</td>
<td>thumbnail</td>
<td>Optional</td>
<td>Specifies maximum width of the thumbnail</td>
</tr>
<tr>
<td>maxlength</td>
<td>thumbnail</td>
<td>Optional</td>
<td>Specifies the maximum length of the thumbnail</td>
</tr>
<tr>
<td>maxscale</td>
<td>thumbnail</td>
<td>Optional</td>
<td>Specifies the maximum scale of the thumbnail</td>
</tr>
<tr>
<td>name</td>
<td>addWatermark</td>
<td>deletePages</td>
<td>getInfo</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>newOwnerPassword</td>
<td>protect</td>
<td>Optional (see Description)</td>
<td>Password used to set permissions on a PDF document. To change the default permissions, specify the <code>newOwnerPassword</code> attribute. For more information, see the section PDF document passwords.</td>
</tr>
<tr>
<td>newUserPassword</td>
<td>protect</td>
<td>Optional (see Description)</td>
<td>Password used to open PDF document. Specify either the <code>newUserPassword</code> attribute or the <code>newOwnerPassword</code> attribute; if you specify both, the passwords must differ. For more information, see the section PDF document passwords.</td>
</tr>
<tr>
<td>noattachments</td>
<td>thumbnail</td>
<td>Optional</td>
<td>Removes all attachments from PDF documents.</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>----------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>noattachments</td>
<td>optimize</td>
<td>Optional</td>
<td>Remove all file attachments</td>
</tr>
<tr>
<td>nobookmarks</td>
<td>optimize</td>
<td>Optional</td>
<td>Remove bookmarks from PDF document</td>
</tr>
<tr>
<td>nocomments</td>
<td>optimize</td>
<td>Optional</td>
<td>Remove comments from PDF document</td>
</tr>
<tr>
<td>nofonts</td>
<td>optimize</td>
<td>Optional</td>
<td>Remove font styling</td>
</tr>
<tr>
<td>nojavascripts</td>
<td>optimize</td>
<td>Optional</td>
<td>Remove all document level JavaScript actions</td>
</tr>
<tr>
<td>nolinks</td>
<td>optimize</td>
<td>Optional</td>
<td>Remove external cross-references</td>
</tr>
<tr>
<td>nometadata</td>
<td>optimize</td>
<td>Optional</td>
<td>Remove document information and metadata</td>
</tr>
<tr>
<td>nothumbnails</td>
<td>optimize</td>
<td>Optional</td>
<td>Remove embedded page thumbnails</td>
</tr>
<tr>
<td>numberformat</td>
<td>addfooter</td>
<td>Optional</td>
<td>Specify the numbering format for PDF pages in the footer.</td>
</tr>
<tr>
<td>opacity</td>
<td>addWatermark addheader addfooter</td>
<td>Optional</td>
<td>Opacity of the watermark. Valid values are integers in the range 0 (transparent) through 10 (opaque).</td>
</tr>
</tbody>
</table>
| **order** | **merge** | **Optional** | **Order in which the PDF documents in the directory are merged:**  
|          |          |              | name: orders the documents alphabetically by filename.  
|          |          |              | time: orders the documents by timestamp. By default, ColdFusion merges the files in descending order (for example, from Z to A). To change this, set the ascending attribute to **yes**.  
| **outputFiles** | **processddx** | **Required** | **Structure that contains the output files in the DDX file or string as keys and the pathname to the result file as the value.**  
| **overwrite** | **addSign**  
|          | **addWatermark**  
|          | **archive**  
|          | **deletePages**  
|          | **merge**  
|          | **protect**  
|          | **removeWatermark**  
|          | **setInfo**  
|          | **thumbnail**  
|          | **write**  
|          | **tranform**  
|          | **addheader**  
|          | **addfooter**  
|          | **removeheaderfooter**  
|          | **Optional** | **Specifies whether PDF output overwrites the destination file:**  
|          |          |              | yes: overwrites the destination file.  
|          |          |              | no: does not overwrite the destination file. For the thumbnail action, specifies whether to overwrite the destination directory. If the directory exists, the thumbnails are not generated unless overwrite is set to yes.  
| **package** | **merge** | **Optional** | **Create PDF packages**  
<p>| | | |
|          |          |              |</p>
<table>
<thead>
<tr>
<th><strong>pages</strong></th>
<th><strong>addWatermark</strong></th>
<th><strong>deletePages</strong></th>
<th><strong>merge</strong></th>
<th><strong>removeWatermark</strong></th>
<th><strong>optimize</strong></th>
<th><strong>extracttext</strong></th>
<th><strong>extractimage</strong></th>
<th><strong>sign</strong></th>
<th><strong>addheader</strong></th>
<th><strong>addfooter</strong></th>
<th><strong>removeheaderfooter</strong></th>
<th><strong>transform</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Required:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Optional:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Page or pages in the source PDF document on which to perform the action. You can specify multiple pages and page ranges as follows: &quot;1,6-9,56-89,100,110-120&quot;. For the removeWatermark action, the pages attribute applies only to the watermark type. ColdFusion ignores duplicate pages and numbers greater than the total page count. For action=sign, value of attribute pages should consist of just one page number.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>password</strong></th>
<th><strong>addWatermark</strong></th>
<th><strong>addSign</strong></th>
<th><strong>archive</strong></th>
<th><strong>deletePages</strong></th>
<th><strong>getInfo</strong></th>
<th><strong>merge</strong></th>
<th><strong>protect</strong></th>
<th><strong>read</strong></th>
<th><strong>readsignaturefields</strong></th>
<th><strong>removeWatermark</strong></th>
<th><strong>setInfo</strong></th>
<th><strong>sign</strong></th>
<th><strong>thumbnail</strong></th>
<th><strong>unsign</strong></th>
<th><strong>write</strong></th>
<th><strong>optimize</strong></th>
<th><strong>extracttext</strong></th>
<th><strong>extractimage</strong></th>
<th><strong>addheader</strong></th>
<th><strong>addfooter</strong></th>
<th><strong>removeheaderfooter</strong></th>
<th><strong>validatesignaturefields</strong></th>
<th><strong>transform</strong></th>
<th><strong>validate</strong></th>
<th><strong>validate</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optional</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Owner or user password of the source PDF document, if the document is password-protected.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>permissions</td>
<td>protect</td>
<td>Optional</td>
<td>Type of permissions on the PDF document:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>----------</td>
<td>------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>All</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowAssembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowCopy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowDegradedPrinting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowFillIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowModifyAnnotations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowModifyContents</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowPrinting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowScreenReaders</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowSecure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Except for All or None, you can specify a comma-separated list of permissions. To set permissions, you must also set the newOwnerPassword attribute.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| position | addWatermark sign | Optional | The position attribute is the coordinate of the lower left corner of the signature field. |

| resolution | thumbnail | Optional | Image quality used to generate thumbnail images: |
|           |          |          | high: use high resolution (uses more memory). |
|           |          |          | low: use low resolution. |

<p>| rotation | addWatermark transform | Optional | Degree of rotation of the watermark image on the page, for example, &quot;30&quot;. |</p>
<table>
<thead>
<tr>
<th>saveOption</th>
<th>write</th>
<th>Optional</th>
<th>Save options for the PDF output:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>full: normal save (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>incremental: required to save modifications to a signed PDF document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>linear: for faster display.</td>
</tr>
<tr>
<td>scale</td>
<td>thumbnail</td>
<td>Optional</td>
<td>Size of the thumbnail relative to the source page. The value represents a percentage from 1 through 100.</td>
</tr>
<tr>
<td>showOnPrint</td>
<td>addWatermark</td>
<td>Optional</td>
<td>Specify whether to print the watermark with the PDF document:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes: the watermark is printed with the PDF document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no: the watermark is display-only.</td>
</tr>
<tr>
<td>signaturefieldName</td>
<td>sign</td>
<td>Optional</td>
<td>Name of signature field in which the user wants to sign or the name of the signature field that the user wants to unsign.</td>
</tr>
<tr>
<td></td>
<td>unsigned</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| sign             | unsign       |          |                                  |
| **source** | archive | Required (see Usage section for merge) | PDF document used as the source. The source can be one of the following:
An absolute or relative pathname to a PDF document, for example, c:\work\myPDF.pdf or myPDF.pdf.
A PDF document variable in memory that is generated by the cfdocum ent tag or the cfpdf tag, for example, "myPDFdoc". |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>addWatermark</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>deletePages</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>getInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>merge</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>protect</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>read</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>removeWatermark</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>setInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sign</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>thumbnail</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>write</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>optimize</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>extracttext</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>extractimage</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>addheader</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>addfooter</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>removeheaderfooter</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>unsign</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>transform</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| **stopOnError** | merge | Optional | Valid only if the directory attribute is specified. If the specified directory contains files other than ColdFusion-readable PDF files, ColdFusion either stops merge process or continues.

yes: stops the merge process if invalid PDF files exist in the specified directory.

no: continues the merge process even if invalid files exist in the specified directory. |
|---|---|---|---|

| **transparent** | thumbnail | Optional | (format="png" only)
Specifies whether the image background is transparent or opaque:

yes: the background is transparent.

no: the background is opaque. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Value</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>unsignall</td>
<td>unsign</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unsigns all the signatures in the document.</td>
<td></td>
</tr>
<tr>
<td>useStructure</td>
<td>extracttext</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Let's you extract content based on the PDF structure. For better readability of the extracted text, use this attribute together with the attribute honourspaces.</td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>write</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Version of the PDF used to write the document:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, see the section PDF versions.</td>
<td></td>
</tr>
<tr>
<td>Width</td>
<td>sign</td>
<td>optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Width of the signature field.</td>
<td></td>
</tr>
</tbody>
</table>

⚠️ **Note**

To modify the PDF source document, specify the same file pathname for the source and destination attributes, and set the overwrite attribute to yes.

Archiving a signed PDF created using ColdFusion behaves abnormally when opened using Adobe Acrobat. To fix the document for archiving enable the “Changing the Document” permission under the Security options of Acrobat.

**Usage**

You use the cfpdf tag to manipulate and assemble existing PDF documents. Although the cfpdf tag provides much of the functionality available in Acrobat, you cannot use this tag to generate a PDF document from another file format. To create PDF output from HTML and CFML content, use the cfdocument tag.

You cannot embed a cfpdf tag within a cfdocument tag or embed a cfdocument tag within a cfpdf tag; however, you can write the output of a cfdocument tag to a variable and pass the variable to the cfpdf tag. The following example shows how to use the cfdocument tag to create a cover page and add it to a merged PDF document:
<!--- Use the cfdocument tag to create a cover page and write the output to a variable called cfdoc. --->
<cfdocument format="PDF" name="cfdoc">
<html>
<body>
<h1>Here is a cover page</h1>
</body>
</html>
</cfdocument>

<!--- Use the cfpdf tag and cfpdfparam tags to merge individual PDF documents into a new PDF document called new.pdf. Notice that the cfdoc variable created by using the cfdocument tag is the source value of the first cfpdfparam tag. --->
<cfpdf action="merge" destination="/samtemp/pdfs/new.pdf" overwrite="yes">
  <cfpdfparam source="cfdoc"/>
  <cfpdfparam source="/samtemp/pdfs/pdf2.pdf"/>
  <cfpdfparam source="/samtemp/pdfs/pdf1.pdf"/>
</cfpdf>

You can use the cfpdf tag to assemble interactive PDF form files into a single PDF document and flatten forms created in Acrobat (by using the flatten attribute with the write action); however, to process PDF form data, use the cfpdfform and related tags. You cannot use the cfpdf tag to flatten forms created in Adobe LiveCycle Designer ES.

Reading and writing PDF files

The cfpdf tag provides several options for reading and writing PDF files. You can specify a PDF variable or a PDF file as the source, and you can write the output to a variable or to a file (but not both). The following table explains the read and write operations:

<table>
<thead>
<tr>
<th>Task</th>
<th>Attributes</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overwrite a source PDF file</td>
<td>Specify the PDF file pathname as the source and do not specify a destination.</td>
<td><code>&lt;cfpdf action=&quot;addWatermark&quot; source=&quot;myPDF.pdf&quot; image=&quot;myImage.jpg&quot;</code></td>
</tr>
<tr>
<td>Write a PDF document in memory to a file</td>
<td>Specify the PDF variable as the source and a PDF file pathname for the destination.</td>
<td><code>&lt;cfpdf action=&quot;addWatermark&quot; source=&quot;myPDF&quot; image=&quot;myImage.jpg&quot; destination=&quot;outputFile.pdf&quot;</code></td>
</tr>
<tr>
<td>Write a PDF document to a new file</td>
<td>Specify a PDF file pathname as the source and a different PDF file pathname as the destination.</td>
<td><code>&lt;cfpdf action=&quot;addWatermark&quot; source=&quot;sourceFile.pdf&quot; image=&quot;myImage.jpg&quot; destination=&quot;outputFile.pdf&quot;&gt;</code></td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Write a PDF file to a PDF variable</td>
<td>Specify the PDF file pathname as the source and a PDF variable name.</td>
<td><code>&lt;cfpdf action=&quot;addWatermark&quot; source=&quot;sourceFile.pdf&quot; image=&quot;myImage.jpg&quot; name=&quot;myPDF&quot;&gt;</code></td>
</tr>
<tr>
<td>Overwrite a PDF document in memory</td>
<td>Specify the PDF variable name as the source and do not specify a destination.</td>
<td><code>&lt;cfpdf action=&quot;addWatermark&quot; source=&quot;myPDF&quot; image=&quot;myImage.jpg&quot;&gt;</code></td>
</tr>
</tbody>
</table>

### Working with PDF files in memory

ColdFusion gives you the option to write a PDF file to a variable by using the `name` attribute, which is useful if you want to perform multiple operations on a document before writing it to a file. However, this is practical for small files only because of memory requirements. If you are working with large PDF documents, write the PDF documents to files. ColdFusion recommends that you do not specify the `name` attribute when you specify a variable as the source because it creates a copy, which increases processing. In most cases, this is unnecessary because you can reuse variables even after you write them to files.

**Note**

When you use PDF variables within a try/catch block and ColdFusion generates an error, the variables are unusable after the error is generated.

### Printing PDF documents

Use the `cfprint` tag to print PDF documents. Markups, such as sticky notes, comments, and editorial revisions, are not printed with the document.

- **addWatermark action** Use the `addWatermark` action to add a watermark to specified pages in a PDF document. You can add a watermark in one of the following ways:
  - Use the first page of another PDF document as a watermark. ColdFusion overlays the `copyFrom` page on the source document, without enlarging the image.
  - Specify an image file to use as a watermark.
  - Specify an image in memory by using an image variable.

  The following code shows how to use the first page of a PDF document as a watermark:

  ```
  <cfpdf action="addWatermark" source="c:\myBook.pdf" copyFrom="e:\yourBook.pdf" destination="ourBook.pdf" overwrite="yes">
  ```

  By default, ColdFusion applies the watermark to all of the pages in the output file, with the watermark...
To specify a ColdFusion image as a watermark, use the cfiimage tag or Image functions. The addwatermark action also supports RGB and ARGB images, especially the images added using the cfimageget and related functions. The following example converts an image to grayscale and applies it as a watermark to a PDF file:

```coldfusion
<!--- Use the ImageNew function to create a ColdFusion image from a JPEG file. --->
<cfset myImage=ImageNew("../cfdocs/images/artgallery/jeff05.jpg")>

<!--- Use the ImageGrayscale function to convert the image to grayscale in memory. --->
<cfset ImageGrayscale(myImage)>

<!--- Specify the image variable to apply the grayscale image as a watermark in the Book.pdf file. Because the source and destination are the same and the overwrite attribute is set to yes, ColdFusion overwrites the source file. --->
<cfpdf action="addWatermark" source="Book.pdf" destination="Book.pdf" overwrite="yes" image="#myImage#">
```

For more information on ColdFusion images, see Creating and Manipulating ColdFusion Images in the Developing ColdFusion Applications.

- **addfooter** Use this action to add a footer in a PDF document. Specify the source where the PDF document is located and the destination where the new PDF document with the footer is saved, as shown in the following code snippet:

```coldfusion
<cfpdf action="addfooter"
source="../myBook.pdf"
destination="../myBookwithfooter.pdf"
image="adobelogo.JPG" // Use this attribute to add an image in the footer
align = "right"> // By default, the alignment is center
```

You can also specify an image or text that you have to insert in the footer along with various other attributes such as align, bottommargin, leftmargin, numberformat, and opacity.

- **addheader** Use this action to add header in a PDF document. Specify the source and destination for the PDF document and specify the text or image that you want to insert in the header, as shown in the following code:
• **deletePages action** Use the `deletePages` action to remove pages from a specified PDF document. You can specify a single page, a page range, or a comma-separated list of pages, as the following code shows:

```cfml
<cfpdf action="deletePages" source="c:\myBook.pdf" pages="1,16-32,89,100-147" destination="myLittleBook.pdf">
```

• **extracttext** Use the `extracttext` action to extract all words from the specified page numbers in the PDF document, as shown in the following code snippet:

```cfml
<cfpdf action = "extracttext" source = "../myBook.pdf" pages = "5-20, 29, 80" destination = "../adobe/textdoc.txt"
```

• **extractimage** Use the `extractimage` action to extract all images from the specified page number in a PDF document, as shown in the following code snippet:

```cfml
<cfpdf action = "extractimage" source = "../myBook.pdf" pages = "1-200" destination = "../mybookimages" imageprefix = "mybook"
```

The images are extracted and saved in the directory that you specify in the destination attribute. You can specify a prefix for the images (`imageprefix`) being extracted, otherwise the system prefixes the image name similar to "cf+page number". To save the images in a specific format, use the `format` attribute.

• **getInfo action** Use the `getInfo` action to extract information associated with the PDF document, such as the author, title, and creation date. You specify the name of the structure variable that contains the relevant data associated with the file, as the following code shows:

```cfml
<cfpdf action="getInfo" source="myBook.pdf" name="PDFInfo">
<p><cfoutput>#PDFInfo.title#</cfoutput></p>
<p><cfoutput>#PDFInfo.author#</cfoutput></p>
<p><cfoutput>#PDFInfo.keywords#</cfoutput></p>
<p><cfoutput>#PDFInfo.created#</cfoutput></p>
```

For a complete list of information elements, use the `cfdump` tag, as the following code shows:
Reducing quality of PDF document

The optimize action is used to downsample images and discard unused objects in a PDF document.

- **optimize** To downsample images in a PDF document, the `algos` attribute is used with values `bilinear`, `bicubic`, and `nearest_neighbour`. The following code snippet generates a PDF after image downsampling:

  ```coldfusion
  <cfpdf action = "optimize" algo = "bicubic" source ".\myBook.pdf" name = 
  #mybook#>
  ```

  You can also discard unused objects such as comments, JavaScripts, attachments, bookmarks, and metadata from your PDF document using the following attributes with optimize action:

  ```coldfusion
  <cfpdf action = "optimize"
  noJavaScripts
  noThumbnails
  noBookmarks
  noComments
  noMetadata
  noFileAttachments
  noLinks
  nofonts>
  ```

Transforming pages in a PDF document

You can scale a page, specify the position, and rotation values for pages in a PDF document.

- **transform** The transform action has four attributes that define the size (`hscale`, `vscale`), position (`position`), and rotation (`rotation`) of a page. The following code snippet shows the usage:
The value for rotation must be in steps (0, 90, 180, 270). If you specify any other value, the system generates an error.

**PDF file information elements**

The following table describes the information elements you can retrieve with the `getinfo` action:

<table>
<thead>
<tr>
<th>Element</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Acrobat PDFMaker 7.0.7 for Word</td>
<td>Application used to create the PDF document. This value is read-only.</td>
</tr>
<tr>
<td>Author</td>
<td>Harper Lee</td>
<td>Author of the PDF document. You can specify a text string with the <code>setInfo</code> action.</td>
</tr>
<tr>
<td>CenterWindowOnScreen</td>
<td>empty string</td>
<td>Display setting for initial view of the PDF document. To change this setting, use the <code>processddx</code> action with the <code>InitialViewProfile</code> DX element.</td>
</tr>
<tr>
<td>ChangingDocument</td>
<td>Not Allowed</td>
<td>Permissions assigned for editing the PDF content. To change this setting, use the <code>permissions</code> attribute with the <code>protect</code> action.</td>
</tr>
<tr>
<td>Commenting</td>
<td>Allowed</td>
<td>Permissions assigned for adding comments to the PDF document. To change this setting, use the <code>permissions</code> attribute with the <code>protect</code> action.</td>
</tr>
<tr>
<td>ContentExtraction</td>
<td>Allowed</td>
<td>Permissions assigned for extracting content from the PDF document. To change this setting, use the <code>permissions</code> attribute with the <code>protect</code> action.</td>
</tr>
<tr>
<td>Property</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CopyContent</td>
<td>Allowed</td>
<td>Permissions assigned for copying content from the PDF document. To change this setting, use the permissions attribute with the protect action.</td>
</tr>
<tr>
<td>Created</td>
<td>D:20061121155226-05'00'</td>
<td>System-generated creation date of the PDF document. You can specify a text string with the setInfo action.</td>
</tr>
<tr>
<td>DocumentAssembly</td>
<td>Not Allowed</td>
<td>Permissions assigned for merging the PDF document with other PDF documents. To change this setting, use the permissions attribute with the protect action.</td>
</tr>
<tr>
<td>Encryption</td>
<td>Password Security</td>
<td>Specifies whether the PDF file is password-protected. To change the encryption algorithm, or add a password, use the protect action.</td>
</tr>
<tr>
<td>FilePath</td>
<td>C:\ColdFusion\wwwroot\lion\myDoc.pdf</td>
<td>Absolute pathname for the PDF file. This value is read-only.</td>
</tr>
<tr>
<td>FillingForm</td>
<td>Allowed</td>
<td>Permissions assigned for entering data in form fields. To change this setting, use the permissions attribute with the protect action.</td>
</tr>
<tr>
<td>FitToWindow</td>
<td>empty string</td>
<td>Display setting for initial view of the PDF document. To change this setting use the processddx action with the InitialViewProfile DDX element.</td>
</tr>
<tr>
<td>HideMenubar</td>
<td>empty string</td>
<td>Display setting for initial view of the PDF document. To change this setting, use the processddx action with the InitialViewProfile DDX element.</td>
</tr>
<tr>
<td>HideToolbar</td>
<td>empty string</td>
<td>Display setting for initial view of the PDF document. To change this setting, use the processddx action with the InitialViewProfile DDX element.</td>
</tr>
<tr>
<td>Property</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HideWindowUI</td>
<td>empty string</td>
<td>Display setting for initial view of the PDF document. To change this setting, use the <code>processddx</code> action with the <code>InitialViewProfile</code> DDX element.</td>
</tr>
<tr>
<td>Keywords</td>
<td>marketing, sales, production</td>
<td>Keywords specified for searches in the PDF document. You can specify a comma-separated list of keywords with the <code>setInfo</code> action.</td>
</tr>
<tr>
<td>Language</td>
<td>EN-US</td>
<td>Language version used to create the source file for the PDF document. This value is read-only.</td>
</tr>
<tr>
<td>Modified</td>
<td>D:20061121155226-06'00'</td>
<td>System-generated timestamp for when the PDF file was last modified. You can specify a text string with the <code>setInfo</code> action.</td>
</tr>
<tr>
<td>PageLayout</td>
<td>OneColumn</td>
<td>Display setting for initial view of the PDF document. To change this setting, use the <code>processddx</code> action with the <code>InitialViewProfile</code> DDX element.</td>
</tr>
<tr>
<td>Printing</td>
<td>Allowed</td>
<td>Permissions assigned for printing the document. To change this setting, use the <code>permissions</code> attribute with the <code>protect</code> action.</td>
</tr>
<tr>
<td>Producer</td>
<td>Acrobat Distiller 7.0.5 (Windows)</td>
<td>Version of Acrobat Distiller used to generate the PDF document. This value is read-only.</td>
</tr>
<tr>
<td>Properties</td>
<td>empty string</td>
<td>This value is read-only.</td>
</tr>
<tr>
<td>Secure</td>
<td>Not Allowed</td>
<td>Display setting that shows whether the PDF document is password protected.</td>
</tr>
<tr>
<td>ShowDocumentsOption</td>
<td>empty string</td>
<td>Display setting for initial view of the PDF document. To change this setting, use the <code>processddx</code> action with the <code>InitialViewProfile</code> DDX element.</td>
</tr>
<tr>
<td><strong>ShowWindowsOption</strong></td>
<td><strong>empty string</strong></td>
<td>Display setting for initial view of the PDF document. To change this setting, use the <code>procesddx</code> action with the <code>InitialViewProfile DX</code> element.</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Signing</strong></td>
<td>Allowed</td>
<td>Permissions for allowing electronic signatures to the PDF document. To change this setting, use the <code>permissions</code> attribute with the <code>protect</code> action.</td>
</tr>
<tr>
<td><strong>Subject</strong></td>
<td>Product Marketing</td>
<td>The subject assigned to the PDF document. You can specify a text string with the <code>setInfo</code> action.</td>
</tr>
<tr>
<td><strong>Title</strong></td>
<td>Chapter 1: Getting Started</td>
<td>The title assigned to the PDF document. You can specify a text string with the <code>setInfo</code> action.</td>
</tr>
<tr>
<td><strong>TotalPages</strong></td>
<td>25</td>
<td>Total pages in the PDF document. This value is read-only.</td>
</tr>
<tr>
<td><strong>Trapped</strong></td>
<td><code>empty string</code></td>
<td>Indicates whether trapping is applied to the PDF document. Trapping is used in printing to eliminate gaps between two adjoining ink colors. You can specify a text string with the <code>setInfo</code> action.</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>1.6</td>
<td>Version of the Adobe PDF generator used to create the PDF document. To change this setting use the <code>version</code> attribute with the <code>write</code> action. For more information, see the section PDF versions.</td>
</tr>
</tbody>
</table>

- **merge action** Use the `merge` action to assemble PDF documents or pages from PDF source files into one output file. The following code shows how to merge all the PDF files in a directory:

  ```cfcoldfusion
  <cfpdf action="merge" directory="c:\myPDFfiles" destination="oneBigFile.pdf" overwrite="yes">
  ```

  By default, ColdFusion adds the files in descending order by timestamp. The following code merges the source files in ascending order by filename:
This is useful if the source files have logical names, such as Chap0.pdf, Chap1.pdf, Chap2.pdf, and so on. By default, ColdFusion continues the merge process even if it encounters a file in the specified directory that is not a valid PDF document. To stop the merge process if the directory contains files other than valid PDF documents, set the stopOnError attribute to yes:

```cfcdf```
<cfpdf action="merge" directory="c:\bookfiles" destination="book.pdf" overwrite="yes" order="name" ascending="yes" keepBookmark="yes" stopOnError="yes">
```

To create a PDF file from specific pages in a document, use the source attribute with the pages attribute. The following code creates a file from pages 1-5 of the source document:

```cfcdf```
<cfpdf action="merge" source="myBigBook.pdf" pages="1-5" destination="myShortBook.pdf" overwrite="yes">
```

To merge several files into one document, specify the absolute pathnames of the files in a comma-separated list, as the following code shows:

```cfcdf```
<cfpdf action="merge" source="c:\PDFdocs\myBook\Chap1.pdf, c:\PDFdocs\myBook\Chap2.pdf, c:\PDFdocs\myBook\Chap3.pdf" destination="myBook.pdf" package = "true" overwrite="yes">
```

You can now create PDF packages using the package = "true" attribute with the merge action. For more control over the order of files, to assemble files in different locations, and to extract pages from multiple PDF files, use the cfpdfparam tag with the merge action. For more information on merging PDF files, see Assembling PDF Documents in the Developing ColdFusion Applications. If cfpdf action="merge" and package="yes", all file formats can be used as source. The following sample code has ZIP and JPEG file formats as source:

```cfcdf```
<cfpdf action="merge" package="yes" destination="/myBook/adobetest.pdf" overwrite="yes">
  <cfpdfparam source="/inputFiles/c.zip" />
  <cfpdfparam source="/inputFiles/d.jpg" />
</cfpdf>
```

- **processddx action** Use the processddx action to assemble PDF files by processing Document Description XML (DDX) instructions. DDX is a declarative markup language used by Adobe LiveCycle
Assembler. You can use DDX instructions to perform advanced tasks, such as adding table of contents pages, headers and footers, automatic page numbers, and text-string watermarks to PDF documents. ColdFusion provides a subset of LiveCycle Assembler functionality. To determine whether you can perform your tasks in ColdFusion or whether you have to purchase LiveCycle Assembler, see the tables in the following sections.

For complete DDX syntax, see the Adobe LiveCycle Assembler Document Description XML Reference.

**Supported DDX elements**

The following table lists the DDX elements that ColdFusion supports:

<table>
<thead>
<tr>
<th>About</th>
<th>Author</th>
<th>Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center</td>
<td>DatePattern</td>
<td>DDX</td>
</tr>
<tr>
<td>DocumentInformation</td>
<td>DocumentText</td>
<td>Footer</td>
</tr>
<tr>
<td>Header</td>
<td>InitialViewProfile</td>
<td>Keyword</td>
</tr>
<tr>
<td>Keywords</td>
<td>Left</td>
<td>MasterPassword</td>
</tr>
<tr>
<td>Metadata</td>
<td>NoBookmarks</td>
<td>OpenPassword</td>
</tr>
<tr>
<td>PageLabel</td>
<td>Password</td>
<td>PasswordAccessProfile</td>
</tr>
<tr>
<td>PasswordEncryptionProfile</td>
<td>PDF (see Note)</td>
<td>PDFGroup</td>
</tr>
<tr>
<td>Permissions</td>
<td>Right</td>
<td>StyledText</td>
</tr>
<tr>
<td>StyleProfile</td>
<td>Subject</td>
<td>TableOfContents</td>
</tr>
<tr>
<td>TableOfContentsEntryPattern</td>
<td>TableOfContentsPagePattern</td>
<td>Title</td>
</tr>
<tr>
<td>Watermark</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

⚠️ **Note**

ColdFusion does not support the `certification` and `mergeLayers` attributes of the PDF element.

**Restricted DDX elements**

ℹ️ New in ColdFusion 11 - The following restricted DDX elements are supported in ColdFusion 11 (Enterprise Edition).

If you are using ColdFusion 10 or earlier, the following DDX elements are excluded by ColdFusion:

| ArtBox | AttachmentAppearance | Bookmarks |
You can create DDX instructions in any text editor and save the file with a DDX extension. The following example shows the DDX instructions for merging several documents and generating a table of contents with bookmarks from the source PDF documents:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<DDX xmlns="http://ns.adobe.com/DDX/1.0/"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://ns.adobe.com/DDX/1.0/ coldfusion_ddx.xsd">
  <PDF result="Out1">
    <PDF source="Title"/>
    <TableOfContents/>
    <PDF source="Doc1"/>
    <PDF source="Doc2"/>
    <PDF source="Doc3"/>
  </PDF>
</DDX>
```

Processing DDX instructions in ColdFusion

The following code processes the DDX instructions in ColdFusion:
--- The following code verifies that the DDX file exists and the DDX instructions are valid. --->
<cfif IsDDX("Book.ddx")>

--- The following code maps the PDF source files to the PDF source variables in the DDX file. --->
<cfset inputStruct=StructNew()>
<cfset inputStruct.Title="Title.pdf">
<cfset inputStruct.Doc1="Chap1.pdf">
<cfset inputStruct.Doc2="Chap2.pdf">
<cfset inputStruct.Doc3="Chap3.pdf">

--- The following code maps the PDF output file to the PDF result variable in the DDX file. --->
<cfset outputStruct=StructNew()>
<cfset outputStruct.Out1="output.pdf">

--- The following code process the DDX instructions in the Book.ddx file to generate a merged document. --->
<cfpdf action="processddx" ddxfile="Book.ddx" inputfiles="#inputStruct#" outputfiles="#outputStruct#" name="ddxVar">
<cfelse>
<p>The DDX instructions are not valid.</p>
</cfif>

--- The following code displays a success or failure message. --->
<cfoutput>#ddxVar.Out1#</cfoutput>

--- The following code defines a variable that you use to determine the success or failure of the process. Use the cfoutput tag to display the success or failure message, as the previous example shows, or use the cfdump tag to display a structure:

<cfdump var="#ddxVar#">

This code returns the following information for each output file in the structure:

- "Successful", if the file is assembled successfully.
- "Reason for failure", if the file is not assembled successfully and the reason for failure is known.
- "Failure", if the file is not assembled successfully and the reason for failure is not known.

Use the IsDDX function to determine whether a DDX file or set of instructions is valid.
For detailed examples, see Assembling PDF Documents in the Developing ColdFusion Applications.

- **protect action** Use the protect action to password-protect PDF output files, set permissions, and encrypt PDF output files. When you use the protect action, set a newUserPassword or a newOwnerPassword. (You can set both, as long as the passwords differ.) When you assign a user password to a document, all users must use this password to open the PDF document. The following code adds a user password to a PDF document:
To set the permissions on the output file, set the `newOwnerPassword`. A user who enters the owner password when accessing the PDF file is considered the owner of file. The following example shows how to set a new owner password:

```cfpdf
<cfpdf action="protect" encrypt="AES_128" source="Book.pdf" destination="MysteryBook.pdf" overwrite="yes" newOwnerPassword="pssst" permissions="AllowDegradedPrinting">
```

Because the permissions are set to `AllowDegradedPrinting` in this example, ColdFusion lets users print the document at 150 DPI, but prohibits all other actions. If a user tries to delete the file, for example, ColdFusion generates an error message indicating that the password was entered incorrectly or the permissions do not allow the action to be performed.

ColdFusion does not retain permissions: if you add a `newUserPassword` attribute, you also must set the permission explicitly.

To work with `myVar`, you specify `newownerpw` as the password.

### PDF document passwords

A PDF document can have two kinds of passwords: a *user* password and an *owner* password. The following table describes the two types of ColdFusion passwords and their equivalents in Acrobat:

<table>
<thead>
<tr>
<th>ColdFusion password</th>
<th>Acrobat equivalent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User password</td>
<td>Document Open password, user password</td>
<td>Anyone who tries to open the PDF document must enter the password that you specify. A user password does not allow a user to change restricted features in the PDF document.</td>
</tr>
<tr>
<td>Owner password</td>
<td>Permissions password, master password</td>
<td>Lets the person who enters the password restrict access to features in a PDF document.</td>
</tr>
</tbody>
</table>

When you protect a PDF, your password changes to the one you provide. ColdFusion updates the variable's saved password to the one you provide. However, if you provide both passwords, ColdFusion uses the owner password. The following protects a PDF:

```cfpdf
<cfpdf action="protect" source="myVar" password="oldpassword" permissions="none" newUserPassword="newuserpw" newownerpassword="newownerpw">
```

To get all the properties of the PDF, you do the following:
To get only the properties allowed for the user, you do the following:

```cfpdf action="info" source="myVar" password="newuserpw" name="info">
```

Permissions for PDF documents

The following table lists the permissions an owner can set for PDF documents:

<table>
<thead>
<tr>
<th>Permissions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>There are no restrictions on the PDF document.</td>
</tr>
<tr>
<td>AllowAssembly</td>
<td>Users can add the PDF document to a merged document.</td>
</tr>
<tr>
<td>AllowCopy</td>
<td>Users can copy text, images, and other file content. This setting is required to generate thumbnail images with the <code>thumbnail</code> action.</td>
</tr>
<tr>
<td>AllowDegradedPrinting</td>
<td>Users can print the document at low-resolution (150 DPI).</td>
</tr>
<tr>
<td>AllowFillIn</td>
<td>Users can enter data into PDF form fields. Users can sign PDF forms electronically.</td>
</tr>
<tr>
<td>AllowModifyAnnotations</td>
<td>Users can add or change comments in the PDF document.</td>
</tr>
<tr>
<td>AllowModifyContents</td>
<td>Users can change the file content. Users can add the PDF document to a merged document.</td>
</tr>
<tr>
<td>AllowPrinting</td>
<td>Users can print the document at high-resolution (print-production quality). This setting is required for use with the <code>cfprint</code> tag.</td>
</tr>
<tr>
<td>AllowScreenReaders</td>
<td>Users can extract content from the PDF document.</td>
</tr>
<tr>
<td>AllowSecure</td>
<td>Users can sign the PDF document (with an electronic signature).</td>
</tr>
<tr>
<td>None</td>
<td>Users can view the document only.</td>
</tr>
</tbody>
</table>

Encryption for PDF documents
The `encrypt` attribute sets the type of encryption used for opening a password-protected document. By default, ColdFusion uses the RC4 128-bit encryption algorithm to encrypt PDF files. To change the encryption algorithm, use the `encrypt` attribute with the `protect` action. The following code encrypts the PDF output file with the AES algorithm:

```
<cfpdf action="protect" encrypt="AES_128" source="Book.pdf" destination="MysteryBook.pdf" overwrite="yes" newOwnerPassword="pssst" permissions="AllowDegradedPrinting">
```

ColdFusion supports the following encryption algorithms:

<table>
<thead>
<tr>
<th>Encryption algorithm</th>
<th>Compatibility</th>
<th>Description</th>
</tr>
</thead>
</table>
| AES_128              | Adobe Acrobat 7.0 and later | Advanced Encryption Standard (AES) specifies the Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits. This is the highest encryption level. This encryption algorithm lets users do the following:  
  • Encrypt all document contents.  
  • Encrypt all document contents except for the metadata.  
  • Encrypt only the file attachments. |
| RC4_128M             | Adobe Acrobat 6.0 and later | RC4 specifies the RSA Security software stream cipher for algorithms such as Secure Sockets Layer (SSL), to protect Internet traffic, and WEP, to secure wireless networks. This encryption algorithm lets users do the following:  
  • Encrypt all document contents.  
  • Encrypt all document contents except for the metadata. |
| RC4_128              | Adobe Acrobat 5.0 and later | RC4 128-bit encryption. This encryption algorithm lets users encrypt the document contents, but not the document metadata. |
| RC4_40               | Adobe Acrobat 3.0 and later | RC4 40-bit encryption. This is the lowest encryption level. |
| None                 | -                          | The document is not encrypted. |
Adobe ColdFusion Documentation

Note
Document metadata is used in Internet searches. If the metadata is encrypted, search engines
cannot search the PDF document. Users running an earlier version of Acrobat cannot open a
PDF document with a higher encryption setting. For example, if you specify AES 128 encryption,
a user cannot open the document in Acrobat 6.0 or earlier.
read action Use the read action to read the source PDF document into the namevariable, as the following
code shows:

<cfif IsPDFFile("Book.pdf")>
<cfpdf action="read" source="Book.pdf" name="myBook">
...
</cfif>

removeWatermark action Use the removewatermarkaction to remove a watermark from a PDF document
or specified pages in a document. The following example removes a watermark from the first page of a PDF
document and writes the output to a new file:

<cfpdf action="removeWatermark" source="Book.pdf" pages="1"
destination="newBook.pdf" overwrite="yes">

removeheaderfooter actionUse this action to remove the header and footer from a PDF document or from
specified pages in a document. The following example removes the header and footer from the entire
document:

<cfpdf action = "removeheaderfooter" source="..\mybook.pdf" destination =
"new.pdf">

setInfo action Use the setinfo action to specify information associated with a PDF document to be saved
with it. Create a structure that contains the relevant information. Use the info attribute of the cfpdf tag to
refer to the structure. The following code shows the elements that you can modify by using the setInfoactio
n:

<cfset PDFinfo=StructNew()>
<cfset PDFinfo.Title="Make Way for Ducklings">
<cfset PDFinfo.Author="Donald Duck">
<cfset PDFinfo.Keywords="Huey,Dewy,Louie">
<cfset PDFinfo.Subject="Ducks">
<cfpdf action="setInfo" source="chap1.pdf" info="#PDFinfo#"
destination="meta1.pdf" overwrite="yes">

thumbnail action Use the thumbnail action to generate thumbnail images from the source PDF document.
If you do not specify a destination directory for the thumbnail files, ColdFusion creates a directory for the
thumbnails in the directory where the CFM page is located. If you specify a filename as the source, the

© 2014 Adobe Systems Incorporated. All rights reserved.

948


thumbnail directory name is a concatenation of the name of the source file and _thumbnails. For example, the following code generates a thumbnail image for each page in myBook.pdf and stores them in a directory called myBook_thumbnails:

```<cfpdf action="thumbnail" source="myBook.pdf">```

If the CFM page is located in the directory c:\myProject\genThumbnails.cfm, the pathname for the thumbnails directory is c:\myProject\myBook_thumbnails. By default, ColdFusion generates thumbnail files in JPEG format and the images are scaled to 25% of the original.

You can specify individual pages within the source document to generate thumbnails. Also, you can change the size of the thumbnail; the resolution, the output format (JPEG, PNG, or TIFF); and the prefix used for the thumbnail filenames. The following code generates a low-resolution thumbnail from the first page of the source document that is scaled at 50% of the original size:

```<cfpdf action="thumbnail" source="myBook.pdf" pages="1" destination="/myBook\images/" imagePrefix="Cover" format="png" scale="50" resolution="low"></cfpdf>```

The full output file pathname is as follows:

```
c:\myBook\images\Cover_page_1.png```

**Note**

To generate thumbnail images, the permissions of the source document must include All owCopy. For more information, see Permissions for PDF documents in cfpdf.

With ColdFusion 9, the following new attributes were introduced for the thumbnail action:

- **hires**: You can set this attribute to true to extract high-resolution images from the page. If a document contains high-resolution images and you want to retain the resolution of the images, then this attribute is useful.
  
  For example:

  ```<cfpdf action="thumbnail" source="/WORK/myBook.pdf" destination="/WORK/Testing_CFPDF" overwrite="true" hires="yes">```

- **overridepage**: If you set this attribute to true, the thumbnail generated does not adhere to the PDF page size, but to the image size that is present in that page. If the image is not present, the size is set to the maximum size of the page.

- **compressstiffs**: Use this attribute to compress the size of the thumbnail images. As the name of the attribute suggests, it is only valid for the TIFF format. Following is an example:
<cfpdf action="thumbnail" source="C:\WORK\myBook.pdf" destination="C:\WORK\Testing_CFPDF" overwrite="true" hires="yes" format="tiff" compressstiffs="yes">

• **maxscale**: Use this attribute to specify an integer value for the maximum scale of the thumbnail images.
• **maxlength**: Use this attribute to specify an integer value of the maximum length of the thumbnail images.
• **maxbreadth**: Use this attribute to specify an integer value of the maximum width of the thumbnail. The following example illustrates the use of maxscale, maxlength, and maxbreadth:

```xml
<cfpdf action="thumbnail" source="/WORK/myBook.pdf" destination="/WORK/Testing_CFPDF" overwrite="true" format="jpg" maxscale="3" maxlength="300" maxbreadth="200" hires="yes" scale="100">
```

⚠️ **Note**

Typically, the value of the `scale` attribute is set to 100 when using the `maxscale` attribute.

• **write action** Use the `write` action to write the source PDF document, or the PDF document stored in memory as a variable, to a file. The following code converts a PDF file stored in memory to a different PDF version and writes the output to a new file:

```xml
<cfpdf action="read" source="Book.pdf" name="myBook">
<cfpdf action="write" source="myBook" destination="myBook1.pdf" version="1.4">
```

You can now use either name or destination attributes with the write action. The name attribute takes the value as the PDF document variable. For example, you can write the preceding code snippet as:

```xml
<cfpdf action="read" source="Book.pdf" name="myBook">
<cfpdf action="write" source="myBook" name=#myBook# version="1.4">
```

The new `encodeall` attribute encodes all the unencoded streams in the source. However, it does not discriminate between dumb encodings like LZW and encodings like flate, so only unencoded streams get flate encoded.

⚠️ **Note**

You can now register thumbnail fonts using the font management screen.
PDF versions

Change the PDF version so that users running an older version of Acrobat or Adobe Reader can open the file. The following table shows the compatibility between the PDF version and the corresponding Acrobat and Adobe Reader versions:

<table>
<thead>
<tr>
<th>PDF version</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Acrobat and Adobe Reader 2</td>
</tr>
<tr>
<td>1.2</td>
<td>Acrobat and Adobe Reader 3</td>
</tr>
<tr>
<td>1.3</td>
<td>Acrobat and Adobe Reader 4</td>
</tr>
<tr>
<td>1.4</td>
<td>Acrobat and Adobe Reader 5</td>
</tr>
<tr>
<td>1.5</td>
<td>Acrobat and Adobe Reader 6</td>
</tr>
<tr>
<td>1.6</td>
<td>Acrobat and Adobe Reader 7</td>
</tr>
</tbody>
</table>

To linearize PDF documents for faster web display, set the saveOption attribute to `linear`, as the following code shows:

```
<cfpdf action="write" source="myBook" destination="myBook1.pdf" saveOption="linear" overwrite="yes">
```

Do not use the linear save option if you have to maintain interactivity in PDF forms or if the PDF document is enabled for electronic signatures. To allow for electronic signatures, set the saveOption attribute to `incremental`, as the following code shows:

```
<cfpdf action="write" source="myDraft" destination="mySignedDoc.pdf" saveOption="incremental" overwrite="yes">
```

Use the `flatten` attribute to flatten forms created in Acrobat:

```
<cfpdf action="write" source="myAcrobatForm.pdf" destination="myFlatForm.pdf" flatten="yes" overwrite="yes">
```

**Note**

ColdFusion does not support flattening forms created in Adobe LiveCycle. For more information about forms created in LiveCycle and Acrobat, see [Manipulating PDF Forms in ColdFusion](#) in the Developing ColdFusion Applications.

**Example**
The following example generates thumbnail images from pages in a PDF document and links the thumbnail images to the pages in the PDF document:

```coldfusion
<h3>PDF Thumbnail Demo</h3>

<!--- Create a variable for the name of the PDF document. --->
<cfset mypdf="myBook">
<cfset thisPath=ExpandPath(".")>
<!--- Use the getInfo action to retrieve the total page count for the PDF document. --->
<cfpdf action="getInfo" source="#mypdf#.pdf" name="PDFInfo">
<cfset pageCount="#PDFInfo.TotalPages#">

<!--- Generate a thumbnail image for each page in the PDF source document, create a directory (if it doesn't already exist) in the web root that is a concatenation of the PDF source name and the word "thumbnails", and save the thumbnail images in that directory. --->
<cfpdf action="thumbnail" source="#mypdf#.pdf" overwrite="yes" destination="#mypdf#_thumbnails" scale=60>

<!--- Loop through the images in the thumbnail directory and generate a link from each image to the corresponding page in the PDF document. --->
<cfloop index="LoopCount" from="1" to="#pageCount#" step="1">
  <cfoutput>
    <!--- Click the thumbnail image to navigate to the page in the PDF document. --->
    <a href="#mypdf#.pdf##page=#LoopCount#" target="_blank">
      <img src="#mypdf#_thumbnails/#mypdf#_page_#LoopCount#.jpg"></a>
  </cfoutput>
</cfloop>
```

### Archiving PDF documents

Introduced in ColdFusion 11: Use the action type="archive" to archive the PDF files based on the ISO's PDF/A standards. PDF/A is one set of standards among a suite of PDF-based standards managed by the International Organization for Standardization (ISO). It was developed to enable the long-term preservation of electronic documents and provides specifications for the creation, viewing, and printing of PDF documents, with the intent of preserving final documents of record as self-contained documents. The standard does not define an archiving strategy or the goals of an archiving system. Rather, it identifies a "profile" for a PDF file that makes it possible to reproduce the visual appearance of the document the exact same way in the future. This profile specifies what must be included in the file, while prohibiting features that are not suitable for long-term archiving.
Digitally signing the documents

Introduced in ColdFusion 11: Use the action="sign","unsign","validatesignature","readsessmentfields" for creating a digital signature, removing the digital signature, validating the signatures of a document, and for reading the signature fields respectively.

- For action ="sign", the following attributes are available:
  - **keystore** attribute (Required): The location of the keystore file. For example:C:\OpenSSL\bin\keystore.jks.
  - **keystorepassword** attribute (Required): The password for the keystore.
  - **keyalias** attribute (Optional): Alias of the key with which the certificate and private key are stored in the keystore. If it is not specified, the first entry in the keystore is chosen as the alias.
  - **keystorepassword** attribute (Optional): Password for your private key. If not specified, the keystorepassword is used.
  - **signaturefieldname** attribute (Optional): Fully qualified name of the existing field on which the user wants to sign (or) the signature field which has to be unsigned.

- For action ="sign" and "unsign", the following attribute is available:

**signaturefieldname** attribute (Optional): Fully qualified name of the existing field on which the user wants to sign (or) the signature field which has to be unsigned.
<!---- Sign a specific signature field in a input pdf ---->
<cfpdf action="sign"
    source="#inputfilepath##inputfilename#"
    destination="#signedfilepath##signedfilename#"
    keystore="#certpath#cert.jks" keystorepassword="password"
    overwrite="true" author="true"
    signaturefieldname="sign_me"/>

• For action ="unsign", the following attribute is available:

• **Unsignall** attribute (Optional): If true, then will unsign all signature fields in the source document.

<cfpdf action="unsign"  source="../cfpdf_normalPDFSign.pdf"
    destination="../results/cfpdf_normalPDFSign.pdf" unsignall="true" overwrite="true">

• For action ="validatesignature", the following attribute is available:

• **Name** attribute which will store the resultant structure of this operation. It consists of two elements. One is the
  boolean field which indicates whether all signatures have been deemed valid or not. The second is an array
  list consisting of the names of all the signatures which were invalid.

<cfpdf action="validatesignature"  source="../cfpdf_normalPDFSign.pdf"
    name="pdfInfo">
<cfoutput>#pdfInfo.SUCCESS#</cfoutput>

• For action ="readsignaturefields":

• Reads all the signature fields and returns a query object in the variable
  indicated by attribute name. This query object will have five columns. The five
  columns are:

  a) Signed - fully qualified name of the signature field.
  b) Authored - indicates if this field can contain an author signature or not.
  c) Can_Be_Signed - whether this field can be signed.
  d) Can_Be_Authored - tells if this field is already signed or not
  e) Visible - tells if the field is visible.
<!--- Read info about signature fields in a input pdf in a variable and dump it --->
<cfpdf action="readsignaturefields"
    source="#filepath#inputfilename#" name="signinfo"
    password="owner" />
<cfdump var="#signinfo#"/>
Manipulates existing forms created in Adobe Acrobat and Adobe LiveCycle Designer. The following list describes some of the tasks you can perform with the `cfpdfform` tag:

- Embed an interactive form created in Acrobat LiveCycle in a PDF document. You use the `cfpdfform` tag to embed the PDF form in a `cfdocument` tag.
- Render an existing form created in Acrobat or LiveCycle. This includes prefilling fields from a database or an XML data file and processing form data from an HTTP post or PDF submission.
- Extract or prefill values in stored PDF forms and save the output to a file or use it to update a data source.

History

ColdFusion 8: Added this tag.

Category

**Forms tags**

Syntax

```
populate
<cfpdfform
  required
  action = "populate"
  source = "PDF file pathname|byte array"
  optional
  XMLdata = "XML object|XML string|XML data filename|
  URL that returns XML data"
  destination = "output file pathname"
  overwrite = "yes|no"/
  fdf = "true|false> <!---New attribute that populates data in FDF format instead of
XML with subforms and params--->
  fdfdata = "file name to be imported" <!---- New attribute populates data in FDF format
from the AcroForm--->
read
<cfpdfform
  required
  action = "read"
  source = "pathname|byte array"
  at least one of the following:
  XMLdata = "variable name for XML data"
  result = "structure containing form field values"
  optional
  overwrite = "yes|no"/
  fdfdata = "filename to be exported to"
```
Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also
cfdocument, cfdocumentsection, cfform, cfinput, cfpdf, cfpdfformparam, cfpdfparam, cfpdfsubform, cfprint, IsPDFFile, IsPDFObject, Manipulating PDF Forms in ColdFusion in the Developing ColdFusion Applications.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>NA</td>
<td>Required</td>
<td></td>
<td>Action to perform on the source:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• populate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• read</td>
</tr>
<tr>
<td>destination</td>
<td>populate</td>
<td>Optional</td>
<td>Write to browser</td>
<td>Pathname for the output file. You can specify an absolute pathname or a pathname relative to the context root. The file extension must be PDF or XDP. The file extension determines the format of the file. (The XDP format applies only to LiveCycle forms.) If you do not specify the destination, ColdFusion displays the form in the browser. Do not specify the destination when you embed a form in a PDF document.</td>
</tr>
<tr>
<td>attribute</td>
<td>value</td>
<td>type</td>
<td>default</td>
<td>description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
<td>------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| overwrite          | populate read | Optional   | no      | Specifies whether to overwrite the destination file (if `action="populate"`) or the data file (if `action="read"`):
- yes
- no                                                                                           |
| overwriteData      | populate | Optional   | no      | Specifies whether to overwrite existing data in PDF form fields with data from the data source:
- yes: Overwrite existing data in the form fields with that from the data source.
- no: Retain existing data in form fields and populate only those fields without data.
This attribute applies to data supplied from an XML data source and from the `cfp dfparam` and `cfd ps ubform` tags. |
<p>| result             | read   | Optional(see Description) |         | ColdFusion structure that contains the form field values. Specify the <code>XMLdata</code> attribute or the <code>result</code> attribute; you can specify both. |</p>
<table>
<thead>
<tr>
<th>source</th>
<th>populate</th>
<th>read</th>
<th>Required</th>
<th>Pathname of the source PDF (absolute on-disk or in-memory path, or path relative to the context root) or byte array representing a PDF.</th>
</tr>
</thead>
</table>
| XMLdata | populate | read | Optional (see Description) | Pathname for the XML data file.  
- If action="populate", the data from this file, XML object, or XML string populates the form fields. You can specify a pathname relative to the context root or a relative pathname.  
- If action="read", ColdFusion writes the data to the variable. Specify either the XMLdata attribute or the result attribute for the read action; you can specify both. |
| fdf    | populate | Optional | false | If set to true, the system creates FDF with subforms and params instead of an XML. |
| fdfdata | populateread | Optional | | For populate, you specify the file name from where the FDF data is imported. For read, you specify the file name where the FDF data is exported. |

**Usage**

ColdFusion supports two types of interactive forms: forms created in Adobe Acrobat 6.0 or earlier, and forms
created in Adobe LiveCycle. In Adobe Acrobat Professional and Standard 7.0, Adobe introduced Adobe LiveCycle Designer for creating PDF forms. ColdFusion supports forms created in LiveCycle Designer 7.0 and later. Forms created in Acrobat have a flat structure: a list of fields at the same level. Forms created in LiveCycle Designer are hierarchical, often composed of nested subforms. To map the data to the form field, you use \texttt{cfpdfsubform} tags to recreate the structure of the form in ColdFusion. For examples, see the Usage section of the \texttt{cfpdfsubform} tag and "Manipulating PDF Forms in ColdFusion" in the Developing ColdFusion Applications.

- **populate action** Use the \texttt{populate} action to populate PDF form fields from the specified data file. You can specify a destination to write the output to a file or write the populated form directly to the browser. To display the interactive PDF form in the browser, do not specify a destination.

  The following example shows how to populate a PDF form with an XML data file and display the completed form in a browser:

  ```coldfusion```
  ```
  <cfpdfform source="c:\payslipTemplate.pdf" action="populate"
  XMLdata="c:\formdata.xml"/>
  ```
  ```coldfusion```

  This example shows how to populate a PDF form with an XML data file and write the completed form to a new PDF file:

  ```coldfusion```
  ```
  <!--- Specify an XML file to populate a PDF form. --->
  <cfpdfform source="c:\payslipTemplate.pdf"
  destination="c:\employeeid123.pdf" action="populate"
  XMLdata="c:\formdata.xml"/>
  ```
  ```coldfusion```

  Also, you can specify a URL that returns XML data. In the following example, "http://test1.com/xyz" returns XML content:

  ```coldfusion```
  ```
  <cfpdfform source= "#sourcefile#" action="populate" XMLdata=
  "http://test1.com/xyz" destination="#resultfile#" overwrite="true"/>
  ```
  ```coldfusion```

  For forms created in Acrobat, you can write the output to a PDF file only. For forms created in LiveCycle, you have the option to write the output to an XML Data Package (XDP) file. An XDP file is an XML representation of a PDF file.

  \textbf{Note}

  Supplied values in form fields created in Acrobat or LiveCycle Designer are case sensitive. For example, if a check box in a form requires a "Yes" value, the value "yes" does not populate that field.

  The file extension determines the file format: to save the output in XDP format, use an XDP extension in the destination filename:
You can use one or more `cfpdfformparam` tags within a `cfpdfform` tag to populate individual fields in a PDF form. The following example shows how to populate an existing form created in Acrobat (`payslipTemplate.pdf`) and create a PDF form (`employeeid123.pdf`) with the employeeID and salary fields filled in:

```xml
<cfpdfform source="c:\payslipTemplate.pdf" destination="c:\employeeid123.xdp" action="populate" XMLdata="c:\formdata.xml"/>
```

ColdFusion requires that you reproduce the exact structure of the source PDF form to populate fields. To verify the structure of a PDF form in ColdFusion, use the `read` action of `cfpdfform` tag, and then use the `cfdump` tag to display the result structure. Use a `cfpdfsubform` tag for each level within the structure. For more information, see Manipulating PDF Forms in ColdFusion in the Developing ColdFusion Applications. The following example shows how to populate a form created in LiveCycle. Many forms created from templates in LiveCycle contain a subform called form1. Use the `cfpdfsubform` tag to create a subform in ColdFusion.

```xml
<cfpdfform source="c:\payslipTemplate.pdf" destination="c:\employeeid123.pdf" action="populate">
  <cfpdfsubform name="form1">
    <cfpdfformparam name="employeeId" value="123">
    <cfpdfformparam name="salary" value="$85,000">
  </cfpdfsubform>
</cfpdfform>
```

You can now import files in FDF format using the `populate` action. The following example shows how:

```xml
<cfpdfform source= "write_acrroform.pdf" action="populate" fdfdata="abc.fdf" destination="hello.pdf">
</cfpdfform>
```

If the `fdf` attribute for the populate action is set to `true`, it allows you to populate data in FDF format with subforms and params instead of XML, as shown in the following example:
**read action** Use the read action to read the data from the source PDF form and generate a result structure that contains the form fields and their values. Also, you can use the read action to generate an XML data file from a PDF source file.

The following example shows how to read a PDF file and generate a result structure from the data:

```cfc
<!--- Use the read action to retrieve the values from the saved PDF. --->
<cfpdfform source="c:\employeeid123.pdf" result="resultStruct" action="read"/>
```

You can use the cfdump tag to display the result structure:

```cfc
<cfdump var="#resultStruct#">
```

You can use the result fields in ColdFusion, for example, `#resultStruct.employeeId#` and `#resultStruct.salary#`.

The following example shows how to read a PDF file and write the data to an XML file:
The following example shows how to read a PDF file into a variable that contains XML data:

```cfpdfform source="c:\employeidl23.pdf" XMLdata="myXMLdata" action="read"/>
```

The following example shows how to read a PDF file into an XML data variable and generate a result structure. The `cffile` tag writes the data to an XML file:

```cfset sourcefile = "Grant Application Updated.pdf">
<cfset resultfile = "Expandpath('datafile_result1.xml')">
<!--- Use the `cfpdfform` tag to read data extracted from a form into an XML data variable and generate a result structure. --->
<cfpdfform source="#sourcefile#" action="read" xmldata="#xmldata#" result="#resultstruct#"/>
<!--- Use the `cffile` tag to write the XML data to a file. --->
<cffile action="write"file="#resultfile#" output="#xmldata#">
<!--- Use the `cfdump` tag to display the result structure. --->
<cfdump var="#resultstruct#">
```

### Extracting data from a PDF submission

Use the following code to extract data from a PDF submission and write it to a structure called `fields`:

```cfdump var="#fields#"```  

Use the `cfdump` tag to display the data structure, as follows:

```cfdump var="#fields#"```  

---

**Note**

When you extract data from a PDF submission, always specify `"#PDF.content#"` as the source.

You can set the form fields to a variable, as the following code shows:
Use the populate action of the `cfpdfform` tag to write the output to a file. Specify `"#PDF.content#"` as the source. In the following example, the unique filename is generated from a field on the PDF form:

```html
<cfpdfform action="populate" source="#PDF.content#"
    destination="timesheets\#empForm.txtsheet#.pdf" overwrite="yes"/>
```

Extracting data from an HTTP post submission

An HTTP post submission transmits the data from the PDF form, but not the form itself. You can extract data from the PDF form fields, but you cannot write the output directly to a file. To extract the data and update a database, for example, you must map the fields in the database to the structure and HTTP post data exactly.

**Note**

The structure of the HTTP post data (after submission) is not the same as the structure of the PDF form (before data submission). For examples of both, see Manipulating PDF Forms in ColdFusion in Developing ColdFusion Applications.

To determine the structure of the HTTP post data, use the `cfdump` tag with the form name as the variable to display the data structure, as follows:

```html
<cfdump var="#FORM.form1#"
```

**Note**

When you extract data from an HTTP post submission, always specify the form name as the source. For example, specify `"#FORM.form1#"` for a form generated from a template in LiveCycle Designer. When data extraction that uses the `cfpdfform` tag results in more than one page, instead of returning one structure, ColdFusion returns one structure per page.

Embedding PDF forms within a PDF document

You can use the `cfpdfform` tag inside the `cfdocument` tag to embed an existing interactive PDF form within a PDF document. Use at least one `cfdocumentsection` tag with the `cfpdfform` tag, but do not place the `cfpdfform` tag within the `cfdocumentsection` tag. For more information about embedding PDF forms, see Manipulating PDF Forms in ColdFusion in Developing ColdFusion Applications.

Flattening forms created in Acrobat

You use the `cfpdf` tag to flatten forms created in Acrobat. ColdFusion does not support flattening forms created in LiveCycle. For more information, see Assembling PDF Documents in Developing ColdFusion Applications.

Printing forms
Use the cfprint tag to print forms created in Acrobat. Markups, such as sticky notes, comments, and editorial revisions, are not printed with the form. You cannot use the cfprint tag to print forms created in LiveCycle Designer.

Exporting PDF Forms in FDF

You can export PDF forms in FDF format using the read action. The following example shows how you can export a PDF form in FDF format:

```
<cfpdfform source= "acroform_export.pdf" action="read" fdfdata="abc.fdf" />
```

Example

The following example shows how to embed an interactive PDF form in a PDF document created with the cfdocument tag:
<cfquery name="getEmpInfo" datasource="cfdocexamples">
  SELECT * FROM EMPLOYEES
  WHERE EMAIL = <cfqueryparam value="#form.username#">
</cfquery>

<cfdocument format="pdf">
<cfdocumentitem type="header">
  <font size="-1" align="center"><i>Nondisclosure Agreement</i></font>
</cfdocumentitem>
<cfdocumentitem type="footer">
  <font size="-1"><i>Page <cfoutput>#cfdocument.currentpagenumber# of #cfdocument.totalpagecount#</cfoutput></i></font>
</cfdocumentitem>
</cfdocument>

<!--- The following code creates the first section in the PDF document. --->
<cfdocumentsection>
  <h3>Employee Nondisclosure Agreement</h3>
  <p>Please verify the information in the enclosed form. Make any of the necessary changes in the online form and click the <b>Print</b> button. Sign and date the last page. Staple the pages together and return the completed form to your manager.</p>
</cfdocumentsection>

<!--- The following code embeds an interactive PDF form within the PDF document with fields populated by the database query. The cfpdfiform tag automatically creates a section in the PDF document. Do not embed the cfpdfiform within cfdocumentsection tags. --->
<cfpdfform action="populate" source="c:\forms\embed.pdf">
  <cfpdfsubform name="form1">
    <cfpdfformparam name="txtEmpName" value="#getEmpInfo.FIRSTNAME# #getEmpInfo.LASTNAME#">
    <cfpdfformparam name="txtDeptName" value="#getEmpInfo.DEPARTMENT#">
    <cfpdfformparam name="txtEmail" value="#getEmpInfo.IM_ID#">
    <cfpdfformparam name="txtPhoneNum" value="#getEmpInfo.PHONE#">
    <cfpdfformparam name="txtManagerName" value="Randy Nielsen">
  </cfpdfsubform>
</cfpdfform>

<!--- The following code creates the last document section. Page numbering resumes in this section. --->
<cfdocumentsection>
  <p>I, <cfoutput>#getEmpInfo.FIRSTNAME# #getEmpInfo.LASTNAME#</cfoutput>, hereby attest that the information in this document is accurate and complete.</p>
  <br/>
  <table border="0" cellpadding="20">
  <tr><td width="300">
    <hr />
    <p><i>Signature</i></p>
  </td><td width="150">
    <hr />
    <p><i>Today's Date</i></p>
  </td></tr>
  </table>
</cfdocumentsection>
</cfdocument>
**cfpdfformparam**

**Description**

Provides additional information to the `cfpdfform` tag. The `cfpdfformparam` tag is always a child tag of the `cfpdfform` or `cfpdfsubform` tag. Use the `cfpdfformparam` tag to populate fields in a PDF form.

**History**

ColdFusion 8: Added this tag.

**Category**

*Forms tags*

**Syntax**

```
<cfpdfform ...>
  <cfpdfformparam
      name = "field name"
      value = "ColdFusion variable"
      index = "integer">
  </cfpdfformparam>
</cfpdfform>
```

**Note**

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

**See also**

`cfdocument`, `cfdocumentsection`, `cfform`, `cfinput`, `cfpdf`, `cfpdfform`, `cfpdfparam`, `cfpdfsubform`, `cfprint`, `IsPDFFile`, `IsPDFObject`

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Optional</td>
<td>1</td>
<td>Index associated with the field name. If multiple fields have the same name, use the index value to locate one of them. Applies to forms created in LiveCycle only.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Field name on the PDF form.</td>
</tr>
</tbody>
</table>
value  | Required  | Value associated with the field name. For interactive fields, specify a ColdFusion variable.

Usage

Use the `cfpdfformparam` tag inside the `cfpdfform` tag or the `cfpdfsubform` tag to populate fields in a PDF form. Use the `index` attribute of the `cfpdfformparam` tag to specify fields with the same name and different values, as the following code shows:

```cfm
<!--- This example shows how to use multiple cfpdfformparam tags with the same name and different index values for a PDF form that contains fields with same name. --->
<cfpdfform source="c:\payslipTemplate.pdf" destination="c:\employeeid123.pdf" action="populate">
  <cfpdfformparam name="phone" value="781-869-1234" index="1"/>
  <cfpdfformparam name="phone" value="617-273-9021" index="2"/>
</cfpdfform>
```

**Note**

Use the `index` attribute with forms created in LiveCycle only. Forms created in Acrobat cannot contain more than one field with the same name; therefore the `index` attribute is not valid.

Example

See the `cfpdfform` tag examples.
**cfpdfparam**

**Description**

Provides additional information for the **cfpdf** tag. The **cfpdfparam** tag applies only to the **merge** action of the **cfpdf** tag and is always a child tag of the **cfpdf** tag.

**History**

ColdFusion 8: Added this tag.

**Category**

**Forms tags**

**Syntax**

```
<cfpdf action = "merge" ..>
 <cfpdfparam
       pages = "page number|page range|comma-separated page numbers"
       password = "user or owner password"
       source = "absolute or relative pathname to a PDF file|PDF document variable|
                cfdocument variable">
     </cfpdf>
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfdocument, cfdocumentsection, cfpdf, cfpdfform, cfpdfformparam, cfpdfsubform, cfprint, IsPDF, IsPDFObject

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pages</td>
<td>Optional</td>
<td></td>
<td>Page or pages of the PDF source file to merge. You can specify a range of pages, for example, &quot;1-5&quot;, or a comma-separated list of pages, for example, &quot;1-5,9-10,18&quot;.</td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td></td>
<td>User or owner password, if the source PDF file is password-protected.</td>
</tr>
</tbody>
</table>
source | Required | Source PDF file to merge. You can specify a PDF variable, a `cfdocument` variable, or the pathname to a file.

**Usage**

Use the `cfpdfparam` tag to merge several PDF documents into one file. The `cfpdfparam` tag lets you specify the order of source files explicitly. You can use this tag to merge pages from multiple PDF document source files in different locations.

The following code creates a single PDF document called `combined.pdf` that contains pages 1-3 and page 5 of the file `abc.pdf`, followed by all of the pages in `xyz.pdf`, a file in memory with the variable name `myPDFvariable`, and lastly pages 10-90 from the file `abc.pdf`. The `password` attribute applies only if the source file is password-protected:

```cfpdf
<cfpdf action="merge" destination="combined.pdf" overwrite="yes">
  <cfpdfparam source="c:\abc.pdf" pages="1-3,5" password="adobe">
    \x
  <cfpdfparam source="myPDFvariable">
  <cfpdfparam source="abc.pdf" pages="10-90" password="adobe">
</cfpdf>
```

**Note**

When you use the `cfpdfparam` tag with the `cfpdf merge` action, you must specify either the `destination` attribute or the `name` attribute for the `cfpdf` tag.

**Example**

The following ColdFusion page creates a form for downloading tax forms and tax information booklets:

```html
<h3>Downloading Federal Tax Documents</h3>
<p>Please choose the your type of business.</p>
<!---- Create the ColdFusion form to determine which PDF documents to merge. ---->
<table>
<cfform action="cfpdfMergeAction.cfm" method="post">
  <tr><td><cfinput type="radio" name="businessType" Value="SoleP"> Sole Proprieter</cfinput></td></tr>
  <tr><td><cfinput type="radio" name="businessType" Value="Partner">Partnership</cfinput></td></tr>
  <tr><td><cfinput type="radio" name="businessType" Value="SCorp">S Corporation</cfinput></td></tr>
  <cfinput type = "hidden" name = "selection required" value = "must make a selection">
  <tr><td><cfinput type="Submit" name="OK" label="OK"></td></tr>
</cfform>
</table>
```
The ColdFusion action page merges PDF files in different locations based on the selection in the form:

```coldfusion
<!--- Create a merged PDF document based on the selection in the form. --->
<cfpdf action="merge" name="taxDoc">
  <cfif #form.businessType# is "SoleP">
    <cfpdfparam source="taxForms\f2106ez.pdf">
    <cfpdfparam source="taxForms\f1040.pdf">
    <cfpdfparam source="taxForms\f1040sc.pdf">
    <cfpdfparam source="taxInfo\i1040sc.pdf">
    <cfpdfparam source="taxInfo\i2106.pdf">
    <cfpdfparam source="taxInfo\i1040sc.pdf">
    <cfpdfparam source="taxInfo\p535.pdf">
    <cfpdfparam source="taxInfo\p560.pdf">
    <cfpdfparam source="taxInfo\p334.pdf">
  </cfif>
  <cfelseif #form.businessType# is "Partner">
    <cfpdfparam source="taxForms\f1065.pdf">
    <cfpdfparam source="taxForms\f1065b.pdf">
    <cfpdfparam source="taxForms\f1065bsk.pdf">
    <cfpdfparam source="taxForms\f8804.pdf">
    <cfpdfparam source="taxForms\f8825.pdf">
    <cfpdfparam source="taxInfo\p535.pdf">
    <cfpdfparam source="taxInfo\p560.pdf">
    <cfpdfparam source="taxInfo\p334.pdf">
  </cfelseif #form.businessType# is "SCorp">
    <cfpdfparam source="taxForms\f1120s.pdf">
    <cfpdfparam source="taxForms\f2553.pdf">
    <cfpdfparam source="taxForms\f8453s.pdf">
    <cfpdfparam source="taxForms\f8825.pdf">
    <cfpdfparam source="taxInfo\i1120s.pdf">
    <cfpdfparam source="taxInfo\p542.pdf">
    <cfpdfparam source="taxInfo\p535.pdf">
    <cfpdfparam source="taxInfo\p560.pdf">
  </cfelseif>
</cfpdf>
<cfpdf action="write" source="taxDoc" destination="c:\taxDoc.PDF" overwrite="yes"/>
```

⚠️ **Note**

ColdFusion automatically flattens form fields when you use the `merge` action of the `cfpdf` tag.
**cfpdfsubform**

**Description**

Populates a subform within the *cfpdfform* tag. The *cfpdfsubform* tag can be a child tag of the *cfpdfform* tag or nested in another *cfpdfsubform* tag.

**History**

ColdFusion 8: Added this tag.

**Category**

*Forms tags*

**Syntax**

```
<cfpdfform ..>
  <cfpdfsubform
    name = "field name"
    index = "integer">
    </cfpdfsubform>
</cfpdfform>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfdocument, cfdocumentsection, cfform, cfinput, cfpdf, cfpdfform, cfpdfformparam, cfpdfparam, cfprint, IsPDFFile, IsPDFObject

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Optional</td>
<td>1</td>
<td>Index associated with the field name. If multiple fields have the same name, ColdFusion uses the index value to locate one of them.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Name of the subform corresponding to subform name in the PDF form.</td>
</tr>
</tbody>
</table>

**Usage**
Use the `cfpdfsubform` tag with the `cfpdfform` tag to populate one or more subforms within a PDF form. The `cfpdfsubform` tag can contain multiple `cfpdfformparam` tags. Also, you can nest subforms, as the following example shows:

```xml
<!--- This example shows how to nest cfpdfsubform tags. --->
<cfpdfform source="c:\payslipTemplate.pdf" destination="c:\employeeid123.pdf" action="populate">
  <cfpdfsubform name="employeeDetail">
    <cfpdfsubform name="address">
      <cfpdfformparam name="txtAddLine1" value="572 Evergreen Terrace">
      <cfpdfformparam name="txtCity" value="Springfield">
      <cfpdfformparam name="txtState" value="Oregon">
      <cfpdfformparam name="txtZip" value="65412">
      <cfpdfformparam name="txtCountry" value="United States">
    </cfpdfsubform>
    <cfpdfformparam name="txtEmployeeId" value="879104">
    <cfpdfformparam name="numSalary" value="$85,000">
  </cfpdfsubform>
</cfpdfform>
```

Use subforms to match the exact structure of the source PDF form. If you do not, ColdFusion cannot prefill the form with data and generates an error. Many of the forms generated from templates in LiveCycle contain a subform called form1. You must specify this as a subform in your code, as the following example shows:

```xml
<cfpdfform source="c:\forms\timesheetForm.pdf" action="populate">
  <cfpdfsubform name="form1">
    <cfpdfformparam name="txtCompanyName" value="Adobe">
    <cfpdfformparam name="txtManager" value="Randy Nielsen">
  </cfpdfsubform>
</cfpdfform>
```

To verify the structure of a PDF form in ColdFusion, use the `read` action of the `cfpdfform` tag, as the following example shows:

```xml
<cfpdfform source="c:\forms\timesheetForm.pdf" result="resultStruct" action="read"/>
```

Then use the `cfdump` tag to display the structure:

```xml
<cfdump var="#resultStruct#"/>
```

Example

See the `cfpdfform` tag examples.
**cfpod**

**Description**

Creates a pod, an area of the browser window or layout area with an optional title bar and a body that contains display elements.

**Category**

**Display management tags**

**Syntax**

```xml
<cfpod
    source = "path"
    bodyStyle = "CSS style specification"
    headerStyle = "CSS style specification"
    height = "number of pixels"
    name = "string"
    onBindError = "JavaScript function name"
    title = "string"
    width = "number of pixels"/>
```

OR

```xml
<cfpod
    bodyStyle = "CSS style specification"
    headerStyle = "CSS style specification"
    height = "number of pixels"
    name = "string"
    onBindError = "JavaScript function name"
    title = "string"
    width = "number of pixels">
    pod contents
</pod>
```

If the tag does not have a body and end tag, close it with /> character combination.

⚠️ **Note**

You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.

**See also**

`cfajaximport`, `cfdiv`, `cflayout`, `cfwindow`

**History**

ColdFusion 8: Added this tag.

**Attributes**
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodyStyle</td>
<td>Optional</td>
<td></td>
<td>A CSS style specification for the pod body. As a general rule, use this attribute to set color and font styles. Using this attribute to set the height and width, for example, can result in distorted output.</td>
</tr>
<tr>
<td>headerStyle</td>
<td>Optional</td>
<td></td>
<td>A CSS style specification for the pod header. As a general rule, use this attribute to set color and font styles. Using this attribute to set the height and width, for example, can result in distorted output.</td>
</tr>
<tr>
<td>height</td>
<td>Optional</td>
<td>100</td>
<td>Height if the control, including the title bar and borders, in pixels.</td>
</tr>
<tr>
<td>name</td>
<td>Optional</td>
<td></td>
<td>Name of the pod control.</td>
</tr>
<tr>
<td>onBindError</td>
<td>Optional</td>
<td>See Description</td>
<td>The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and have specified a global error handler (by using the <code>ColdFusion.s etGlobalErrorHandler</code> function), it displays the error message; otherwise a default error pop-up displays.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>-------------</td>
<td></td>
</tr>
</tbody>
</table>
| **overflow** | Optional | Specifies how to display child content whose size would cause the control to overflow the pod boundaries. The following values are valid:  
- **auto**: shows scrollbars when necessary.  
- **hidden**: does not allow access to overflowing content.  
- **scroll**: always shows horizontal and vertical scroll bars, even if they are not needed.  
- **visible**: content can display outside the bounds of the pod.  
**Note**: In Internet Explorer, pods with the **visible** setting expand to fit the size of the contents, rather than having the contents extend beyond the layout area. |
| **source** | Required if the tag does not have a body | A URL that returns the pod contents. ColdFusion uses standard page path resolution rules. If you specify this attribute and the **cfpod** tag has a body, ColdFusion ignores the body contents. You can use a bind expression with dependencies in this attribute; for more information see Usage.  
**Note**: If a CFML page specified in this attribute contains tags that use AJAX features, such as **cform**, **cfgrid**, and **cfwindow**, you must use a **cfajaximport** tag on the page with the **cfpod** tag. For more information, see [cfajaximport](http://docs.adobe.com/cn/en/coldfusion/coldfusion-reference/cfajaximport.html). |
<table>
<thead>
<tr>
<th>title</th>
<th>Optional</th>
<th>Text to display in the pod's title bar. You can use HTML mark-up to control the title appearance, for example to show the text in red italic font. If you omit this attribute, the pod does not have a title bar.</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>Optional</td>
<td>500</td>
</tr>
</tbody>
</table>

Usage

You use a `source` attribute or a tag body to specify the pod contents; if you specify both, ColdFusion uses the contents specified by the `source` attribute and ignores the tag body. If you use a `source` attribute, an animated icon and the text "Loading..." appears while the contents is being fetched.

If the `source` attribute specifies a page that defines JavaScript functions, the function definitions on that page must have the following format:

```
functionName = function(arguments) {function body}
```

Function definitions that use the following format may not work:

```
function functionName (arguments) {function body}
```

However, Adobe recommends that you include all custom JavaScript in external JavaScript files and import them on the application's main page, and not write them inline in code that you get using the `source` attribute. Imported pages do not have this function definition format restriction.

If you use the `source` attribute, you can use a `bind expression` to include form field values or other form control attributes as part of the source specification. You can bind to HTML format form controls only. To use a bind expression, specify a URL and pass one or more URL parameters the page, including `bind parameters`.

In its most basic form, a bind parameter consists of the `name` or `id` attribute of the control to which you are binding in braces ({}). To include the value of the `city` control as a bind parameter, for example, use the following format:

```
source="/myapplication/cityPod.cfm?cityname={city}"`
```

For detailed information about using bind expressions, see Binding data to form fields in the Developing ColdFusion Applications.
### Example

The following CFML page displays two pods in a vertical layout. Each pod gets its contents from a `displayforpod.cfm` page that uses the `cffeed` tag to get an Atom feed.

```html
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Untitled Document</title>
</head>

<body>
<cflayout type="hbox" style="background-color:##CCffFF; color:red;">
  <cflayoutarea>
    <cfpod name="pod01" source="displayforpod.cfm?start=1" height="500" width="300"
           title="Comment 1"/>
  </cflayoutarea>
  <cflayoutarea>
    <cfpod name="pod02" source="displayforpod.cfm?start=2" height="500" width="450"
           title="Comment 2"/>
  </cflayoutarea>
</cflayout>
</body>
</html>
```

The following code shows the contents of the `displayforpod.cfm` page:

```cfml
<cffeed action="read" source="http://googleblog.blogspot.com/atom.xml" query="feedQuery" properties="feedMetadata" >
  <cfloop query = "feedQuery"
     startRow = "#url.start#" endRow = "#url.start#">
    <cfoutput>#feedQuery.content#<br />
    =======================================<br />
  </cfoutput>
</cfloop>
```

© 2014 Adobe Systems Incorporated. All rights reserved.
**cfpop**

**Description**

Retrieves or deletes e-mail messages from a POP mail server.

**Category**

Communications tags, Internet protocol tags

**Syntax**

```cfpop
<cfpop
    server = "server name"
    action = "getHeaderOnly|getAll|delete|markRead"
    attachmentPath = "path"
    debug = "yes|no">
    generateUniqueFilenames = "yes|no"
    messageNumber = "number"
    name = "query name"
    password = "password"
    port = "port number"
    secure = "yes|no"
    startRow = "number"
    timeout = "seconds"
    uid = "number"
    delimeter = "delimeter char"
    username = "user name">
</cfpop>
```

**Note**

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

**See also**

cfftp, cfhttp, cfldap, cfmail, cfmailparam, SetLocale; Sending and Receiving E-Mail in the Developing ColdFusion Applications

**History**

ColdFusion 11: New action, markRead is supported. New attribute delimeter.

ColdFusion 10: Added attribute {{secure}}

ColdFusion MX 7.01: Added cids query variable.

ColdFusion MX 6.1:

- Added support for multipart mail messages with Text and HTML parts.
- Changed the attachment name separator: the TAB character is now the separator between attachment names in the attachments and attachmentfiles query fields if a message has multiple attachments.
This behavior is identical to ColdFusion 5 and earlier versions.

ColdFusion MX: Changed the attachment name separator: the comma separates names in the `attachment` and `attachmentfiles` query fields if a message has multiple attachments.

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>Required</td>
<td></td>
<td>POP server identifier:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• A host name, for example, &quot;biff.upperlip.com&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• An IP address, for example, &quot;192.1.2.22&quot;.</td>
</tr>
<tr>
<td>action</td>
<td>Optional</td>
<td><code>getHeaderOnly</code></td>
<td>• <code>getHeaderOnly</code>: returns message header information only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <code>getAll</code>: returns message header information, message text, and attachments if <code>attachmentPath</code> is specified</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <code>delete</code>: deletes messages on POP server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <code>markRead</code>: marks the message as read</td>
</tr>
<tr>
<td>attachmentPath</td>
<td>Optional</td>
<td></td>
<td>If <code>action=&quot;getAll&quot;</code>, specifies a directory in which to save any attachments. If the directory does not exist, ColdFusion creates it. If you omit this attribute, ColdFusion does not save any attachments. If you specify a relative path, the path root is the ColdFusion temporary directory, which is returned by the <code>GetTempDirectory</code> function.</td>
</tr>
</tbody>
</table>
### debug

- **Optional**
- **Value**:
  - **no**: sends debugging output to standard output. By default, if the console window is unavailable on server configurations, ColdFusion sends output to `cf_root/runtime/logs/coldfusion-out.log`.
  - **yes**: does not generate debugging output.

### generateUniqueFileNames

- **Optional**
- **Value**:
  - **no**: does not generate unique filenames for files attached to an e-mail message, to avoid naming conflicts when files are saved.
  - **yes**: generate unique filenames for files attached to an e-mail message.

### maxRows

- **Optional**
- **Value**:
  - Retrieves all available rows
  - Number of messages to return or delete, starting with the number in `startRow`. Ignored if `messageNumber` or `uid` is specified.

### messageNumber

- **Optional**
- **Value**:
  - Message number or comma-separated list of message numbers to get or delete. Invalid message numbers are ignored. Ignored if `uid` is specified.

### name

- **Required if action = "getAll" or "getHeaderOnly"**
- **Value**:
  - Name for query object that contains the retrieved message information.

### password

- **Optional**
- **Value**:
  - Password that corresponds to `username`.

### port

- **Optional**
- **Value**:
  - 110
  - POP port.

### secure

- **Optional**
- **Value**:
  - **no**: if set to `yes`, enables SSL for pop requests.
### Usage

The **cfpop** tag retrieves one or more mail messages from a POP server and populates a ColdFusion query object with the resulting messages, one message per row. Alternatively, it deletes one or more messages from the POP server.

**Note**

When the **cfpop** tag encounters malformed mail messages, it does not generate errors; instead, it returns empty fields.

To optimize performance, two retrieve options are available. Message header information is typically short, and therefore quick to transfer. Message text and attachments can be long, and therefore take longer to process.

**The attachmentpath attribute**

Use the following syntax to specify an in-memory directory in the **attachmentpath** attribute. In-memory files speed processing of transient data.

```
ram:///path
```

The path can include multiple directories, for example `ram:///petStore/mail/attachments`. You must create the directories in the path before you specify the file. For more information on using in-memory files, see *Working with in-memory files* in *Optimizing ColdFusion applications* in the *Developing ColdFusion Applications*.  

---

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>startRow</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>timeout</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>uid</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>username</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>delimiter</strong></td>
<td>Optional</td>
</tr>
</tbody>
</table>
The cfpop query variables

The following table describes the variables that provide information about the query that is returned by cfpop:

<table>
<thead>
<tr>
<th>Variable names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryname.recordCount</td>
<td>Number of records returned by query.</td>
</tr>
<tr>
<td>queryname.currentRow</td>
<td>Current row that cfoutput is processing.</td>
</tr>
<tr>
<td>queryname.columnList</td>
<td>List of column names in query.</td>
</tr>
</tbody>
</table>

Query message header and body columns

The following table lists the message header and body columns that are returned if action = "getHeaderOnly" or "getAll":

<table>
<thead>
<tr>
<th>Column name</th>
<th>getHeaderOnly returns</th>
<th>getAll returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryname.date</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.from</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.messagenumber</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.messageid</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.replyto</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.subject</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.cc</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.to</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.body</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.textBody</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.HTMLBody</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.header</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.attachments</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.attachmentfiles</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>queryname.UID</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
If the mail message includes a part with a Content-Type of text/plain, the queryname.textBody column contains the part's message content. If the mail message includes a part with a Content-Type of text/HTML, the queryname.HTMLBody column contains the part's message content. If no Content-Type matches these types, the columns are empty. The queryname.Body column always contains the first message body found.

The queryname.attachments column contains a tab-separated list of all the attachment names. The queryname.attachmentfiles column contains a tab-separated list of the locations of the attachment files. Use the `cffile` tag to delete these temporary files when you have processed them.

To create a ColdFusion date/time object from the date-time string that is extracted from a mail message in the queryname.date column, use the following table:

<table>
<thead>
<tr>
<th>Locale</th>
<th>How to create a ColdFusion date/time object from queryname.date</th>
</tr>
</thead>
<tbody>
<tr>
<td>English (US)</td>
<td>Use the <code>ParseDateTime</code> function. If you specify the <code>pop-conversion</code> attribute, the function adjusts the date/time object to UTC.</td>
</tr>
<tr>
<td>Other</td>
<td>Extract the date part of string; pass it to the <code>LSParseDateTime</code> function.</td>
</tr>
</tbody>
</table>

⚠️ **Note**

To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

For more information on `cfpop`, see [Sending and Receiving E-Mail](#) in the Developing ColdFusion Applications.

Example
This view-only example shows the use of `cfpop`.  

`cfpop` lets you retrieve and manipulate mail in a POP3 mailbox. This view-only example shows how to create one feature of a mail client, to display the mail headers in a POP3 mailbox.

To execute this, un-comment this code and run with a mail-enabled CF Server.

```coldfusion
<cfif IsDefined("form.server")>
  <cfif form.server is not "" and form.username is not ""
  <cfpop server = "#form.popserver# " username = #form.username# password = #form.pwd#
    action = "getHeaderOnly" name = "GetHeaders">
  <h3>Message Headers in Your Inbox</h3>
  <p>Number of Records: <cfoutput>#GetHeaders.recordCount#</cfoutput></p>
  <ul>
    <cfoutput query = "GetHeaders">
      <li>Row: #currentRow#: From: #From# -- Subject: #Subject#</li>
    </cfoutput>
  </ul>
</cfif>
</cfif>

<form action = "cfpop.cfm " method = "post">
  Enter your mail server:<p>
  <input type = "Text" name = "popserver"></p>
  Enter your username:<p>
  <input type = "Text" name = "username"></p>
  Enter your password:<p>
  <input type = "password" name = "pwd"></p>
  <input type = "Submit" name = "get message headers"></p>
</form>
```
**cfpresentation**

**Description**

Defines the look of a dynamic slide presentation and determines whether to write the presentation files to disk. The `cfpresentation` tag is the parent tag for one or more `cfpresentationslide` tags, where you define the content for the presentation, and the `cfpresenter` tags, which provide information about the people presenting the slides.

**History**

ColdFusion 9: Added `format` and `destination` attributes.

ColdFusion 8: Added this tag.

**Category**

Data output tags

**Syntax**

```xml
<cfpresentation
   title = "text string"
   authPassword = "authentication password"
   authUser = "authentication user name"
   autoPlay = "yes|no"
   backgroundColor = "hexadecimal color|HTML named color"
   control = "normal|brief"
   controlLocation = "right|left"
   destination = "filepath"
   directory = "pathname"
   format = "ppt|html"
   glowColor = "hexadecimal color|HTML named color"
   initialTab = "outline|search|notes"
   lightColor = "hexadecimal color|HTML named color"
   loop = "yes|no"
   overwrite = "yes|no"
   primaryColor = "hexadecimal color|HTML named color"
   proxyHost = "IP address or server name for proxy host"
   proxyPassword = "password for the proxy host"
   proxyPort = "port of the proxy host"
   proxyUser = "user name for the proxy host"
   shadowColor = "hexadecimal color|HTML named color"
   showNotes = "yes|no"
   showOutline = "yes|no"
   showSearch = "yes|no"
   textColor = "hexadecimal color|HTML named color"
   userAgent = "HTTP user agent identifier">
   presentation content...
</cfpresentation>
```
You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

See also

cfchart, cfpresentationslide, cfpresenter, cfreport, Creating Slide Presentations in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authPassword</td>
<td>Optional</td>
<td></td>
<td>Sends a password to the target URL for Basic Authentication. Combined with <code>username</code> to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerberos authentication.</td>
</tr>
<tr>
<td>authUser</td>
<td>Optional</td>
<td></td>
<td>Sends a user name to the target URL for Basic Authentication. Combined with <code>password</code> to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerberos authentication.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Required</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>autoPlay</td>
<td>Optional</td>
<td>yes</td>
<td>Specifies whether to play the presentation automatically:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• yes: the presentation automatically runs through the entire presentation at startup.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no: the user must click the Play button to start the presentation and click the Next button to advance to the next slide in the presentation.</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>Optional</td>
<td>727971</td>
<td>Background color of the presentation. The value is hexadecimal: use the form &quot;##xxxxxxx&quot; or &quot;##xxxxx xxxxx&quot;, where x = 0-9 or A-F; use two number signs or none. Also, you can use a subset of HTML named colors listed in the section Named colors.</td>
</tr>
<tr>
<td>control</td>
<td>Optional</td>
<td>normal</td>
<td>Presentation control:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• brief</td>
</tr>
<tr>
<td>controlLocation</td>
<td>Optional</td>
<td>right</td>
<td>Specifies the location of the presentation control:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• left</td>
</tr>
<tr>
<td>destination</td>
<td>Optional</td>
<td></td>
<td>Absolute file name or a file path relative to the CFM page. You can use this for both connect presentation and ppt presentations. Required if <code>format=&quot;html&quot;</code>.</td>
</tr>
</tbody>
</table>
| **directory** | Optional | Directory where the presentation is saved. This can be an absolute path or a path relative to the CFM page. Also, ColdFusion creates a subdirectory called `data` that contains:
|              |          | • A SWF file for each slide
|              |          | • `srchdata.xml` (which creates the search interface)
|              |          | • `vconfig.xml`
|              |          | • `viewer.xml`
|              |          | • images, video clips, and SWF files referenced by the `cfpresentationslide` tags
|              |          | If you do not specify a directory, ColdFusion writes the files to a temp directory and runs the presentation in the client browser. |
| **format**   | Optional | Specifies the file format for conversion:
|              |          | • `ppt` converts html input provided in `cfpresentationslide` to a PowerPoint file.
<p>|              |          | • <code>html</code> converts <code>ppt</code> to an HTML presentation. |
| <strong>glowColor</strong>| Optional | 35D334 | Color used for glow effects on the buttons. The value is hexadecimal: use the form &quot;##xxxxxx&quot; or &quot;##xxxxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none. Also, you can use a subset of HTML named colors listed in the section Named colors. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialTab</td>
<td>Optional</td>
<td>outline</td>
<td>Specifies which tab displays on top when the presentation is run. This applies only when the control value is normal:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• outline</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• search</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• notes</td>
</tr>
<tr>
<td>lightColor</td>
<td>Optional</td>
<td>4E5D60</td>
<td>Light color used for light-and-shadow effects. The value is hexadecimal: use the form &quot;##xxxxxxx&quot; or &quot;##xxxxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none. Also, you can use a subset of HTML named colors listed in the section Named colors.</td>
</tr>
<tr>
<td>loop</td>
<td>Optional</td>
<td>no</td>
<td>Specifies whether the presentation runs in a loop:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• yes: the presentation restarts automatically after it ends.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no: the user must click the Play button to restart the presentation.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Optional</td>
<td>yes</td>
<td>Specifies whether files in the directory are overwritten. Valid only when the directory attribute is specified:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• yes: overwrites files if they are already present</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no: creates new files</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>primaryColor</td>
<td>Optional</td>
<td>Primary color of the presentation. The value is hexadecimal: use the form &quot;##xxxxxx&quot; or &quot;##xxxxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none. Also, you can use a subset of HTML named colors listed in the section Named colors.</td>
<td></td>
</tr>
<tr>
<td>proxyHost</td>
<td>Optional</td>
<td>Host name or IP address of a proxy server to which to send the request.</td>
<td></td>
</tr>
<tr>
<td>proxyPassword</td>
<td>Optional</td>
<td>Password required by the proxy server.</td>
<td></td>
</tr>
<tr>
<td>proxyPort</td>
<td>Optional</td>
<td>The port to connect to on the proxy server.</td>
<td></td>
</tr>
<tr>
<td>proxyUser</td>
<td>Optional</td>
<td>User name to provide to the proxy server.</td>
<td></td>
</tr>
<tr>
<td>shadowColor</td>
<td>Optional</td>
<td>Shadow color used for light-and-shadow effects. The value is hexadecimal: use the form &quot;##xxxxxx&quot; or &quot;##xxxxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none. Also, you can use a subset of HTML named colors listed in the section Named colors.</td>
<td></td>
</tr>
<tr>
<td>showNotes</td>
<td>Optional</td>
<td>Specifies whether the Notes tab is present in the presentation control panel:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no</td>
<td></td>
</tr>
<tr>
<td>showOutline</td>
<td>Optional</td>
<td>Specifies whether the Outline is present in the presentation control panel:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>showSearch</td>
<td>Optional</td>
<td>yes</td>
<td>Specifies whether the Search tab is present in the presentation control panel:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no</td>
</tr>
<tr>
<td>textColor</td>
<td>Optional</td>
<td>FFFFFF</td>
<td>Color for all the text in the presentation user interface. The value is hexadecimal: use the form &quot;##xxxxxx&quot; or &quot;##xxx xxxx&quot;, where x = 0-9 or A-F; use two number signs or none. Also, you can use a subset of HTML named colors listed in the section Named colors.</td>
</tr>
<tr>
<td>title</td>
<td>Required</td>
<td></td>
<td>Title of the presentation</td>
</tr>
<tr>
<td>userAgent</td>
<td>Optional</td>
<td>ColdFusion</td>
<td>Text to put in the HTTP User-Agent request header field. Used to identify the request client software.</td>
</tr>
</tbody>
</table>

**Usage**

Use the `<cfpresentation>` tag to create the container for a slide presentation. You can define the position and appearance of the presentation controls, the background color, and the text for the presentation. Also, use this tag to determine whether to write the presentation to files or to run it directly in the client browser. The settings in the `cfpresentation` tag do not affect the appearance of the content defined in the `cfpresentation` tags.

**destination attribute**

Use the following syntax to specify an in-memory file, which is not written to disk in the `destination` attribute. In-memory files speed processing of transient data.

```
ram://filepath
```

The filepath can include directories, for example `ram://petStore/presentations/quarterlyresults.html`. Create the directories in the path before you specify the file. For more information on using in-memory files, see *Working with in-memory files* in *Optimizing ColdFusion applications* in the *Developing ColdFusion Applications*.

**Named colors**

The `cfpresentation` tag supports the following named colors for use with the `backgroundColor`, `glowColor`,...
lightColor, primaryColor, shadowColor, and textColor attributes:

<table>
<thead>
<tr>
<th>Named color</th>
<th>Hexadecimal value</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>FF0000</td>
</tr>
<tr>
<td>green</td>
<td>008000</td>
</tr>
<tr>
<td>blue</td>
<td>0000FF</td>
</tr>
<tr>
<td>black</td>
<td>000000</td>
</tr>
<tr>
<td>white</td>
<td>FFFFFF</td>
</tr>
<tr>
<td>yellow</td>
<td>FFFF00</td>
</tr>
<tr>
<td>gray</td>
<td>808080</td>
</tr>
<tr>
<td>darkgray</td>
<td>A9A9A9</td>
</tr>
<tr>
<td>lightgray</td>
<td>D3D3D3</td>
</tr>
<tr>
<td>cyan</td>
<td>00FFFF</td>
</tr>
<tr>
<td>magenta</td>
<td>FF00FF</td>
</tr>
<tr>
<td>orange</td>
<td>FFA500</td>
</tr>
<tr>
<td>pink</td>
<td>FFC0CB</td>
</tr>
</tbody>
</table>

Example
<!---- This example shows how to create a slide presentation from an HTML file and from HTML code on the CFM page and write the presentation files to a directory called myPresentation, which is relative to the CFM page.---->
<cfpresentation title="Sales Presentation" directory="myPresenation">
  <cfpresenter name="Shyam" title="Vice President" email="shyam@somecompany.com" image="shyam.jpg">
    <cfpresenter name="Ram" title="Sr. Vice President" email="ram@somecompany.com">
      <!--- The following code creates a slide from an HTML file located on the ColdFusion server. --->
      <cfpresentationslide src="introduction.htm" title="Introduction" presenter="Shyam" audio="myAudio.mp3" duration="36"/>

      <!--- The following code creates a slide from HTML code in the CFM file. --->
      <cfpresentationslide>
        <h3>Sales</h3>
        <ul>
          <li>Overview</li>
          <li>Q1 Sales Figures</li>
          <li>Projected Sales</li>
          <li>Competition</li>
          <li>Advantages</li>
          <li>Long Term Growth</li>
        </ul>
      </cfpresentationslide>

      <!--- The following code creates a slide from HTML and CFML code. --->
      <cfpresentationslide title="Q1 Sales Figures" duration="14" presenter="Ram" audio="myAudio2.mp3">
        <h3>Q1 Sales Figures</h3>
        <cfchart format="png" showborder="yes" chartheight="250" chartwidth="300" pieslicestyle="sliced">
          <cfchartseries type="pie">
            <cfchartdata item="Europe" value="9">
            <cfchartdata item="Asia" value="20">
            <cfchartdata item="North America" value="50">
            <cfchartdata item="South America" value="21">
          </cfchartseries>
        </cfchart>
      </cfpresentationslide>
    </cfpresenter>
  </cfpresentation>
cfpresentationslide

Description

Creates a slide dynamically from a source file or HTML and CFML code on the ColdFusion page. The cfpresentationslide is a child tag of the cfpresentation tag.

History

ColdFusion 9: Added the slides attribute. Added PowerPoint file support to the src attribute.

ColdFusion 8: Added this tag.

Category

Data output tags

Syntax

```xml
<cfpresentation ...>
  <cfpresentationslide
    advance = "auto|never|click"
    audio = "pathname relative to the CFM page or the web root for audio file"
    authPassword = "authentication password"
    authUser = "authentication user name"
    duration = "duration of slide in seconds"
    marginBottom = "margin in pixels"
    marginLeft = "margin in pixels"
    marginRight = "margin in pixels"
    marginTop = "margin in pixels"
    notes = "text string"
    presenter = "presenter name"
    scale = "decimal"
    src = "absolute path|URL|path relative to CFM page"
    title = "text string"
    userAgent = "HTTP user agent identifier"
    video = "pathname relative to the CFM page or the web root for video file"
    useExternalProgram = "true|false" />
</cfpresentation>
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfchart, cfpresentation, cfpresenter, cREPORT, Creating Slide Presentations in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| advance     | Optional| See Description | Overrides the `<cfpresentation>` tag autoPlay attribute for the slide:  
  - **auto**: after the slide plays, the presentation advances to the next slide automatically. This is the default value if `cfpresentation autoPlay="yes"`.  
  - **never**: after the slide plays, the presentation does not advance to the next slide until the user clicks the Next button. This is the default value if `cfpresentation autoPlay="no"`.  
  - **click**: after the slide plays, the presentation advances to the next slide if the user clicks anywhere in the main presentation area. |
<p>| audio       | Optional| Pathname of the audio file relative to the CFM page or the web root. The audio file must be an MP3 file. You cannot specify both audio and video for a slide. |
| authPassword| Optional| Use to pass a password to the target URL for Basic Authentication. Combined with <code>username</code> to form a base64 encoded string that is passed in the <code>Authenticate</code> header. Does not provide support for Integrated Windows, NTLM, or Kerberos authentication. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authUser</td>
<td>Optional</td>
<td></td>
<td>Use to pass a user name to the target URL for Basic Authentication. Combined with password to form a base64 encoded string that is passed in the Authenticate header. Does not provide support for Integrated Windows, NTLM, or Kerberos authentication.</td>
</tr>
<tr>
<td>duration</td>
<td>Optional</td>
<td></td>
<td>Duration in seconds that the slide is played. If you do not specify a duration, the slide plays for the duration of the audio clip associated with the slide.</td>
</tr>
<tr>
<td>marginBottom</td>
<td>Optional</td>
<td>0</td>
<td>Bottom margin of the slide.</td>
</tr>
<tr>
<td>marginLeft</td>
<td>Optional</td>
<td>0</td>
<td>Left margin of the slide.</td>
</tr>
<tr>
<td>marginRight</td>
<td>Optional</td>
<td>0</td>
<td>Right margin of the slide</td>
</tr>
<tr>
<td>marginTop</td>
<td>Optional</td>
<td>0</td>
<td>Top margin of the slide</td>
</tr>
<tr>
<td>notes</td>
<td>Optional</td>
<td></td>
<td>Notes used for the slide. Notes are displayed only if the showNotes attribute of the cfPresentationSlide tag is set to yes.</td>
</tr>
<tr>
<td>presenter</td>
<td>Optional</td>
<td></td>
<td>Presenter of the slide. A slide can have only one presenter. This name must match one of the presenter names in the cf presenter tag.</td>
</tr>
<tr>
<td>scale</td>
<td>Optional</td>
<td>1.0</td>
<td>Scale used for the HTML content in the slide presentation. If you do not specify the scale, ColdFusion automatically scales the content to fit in the slide.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>slides</td>
<td>Optional</td>
<td>All slides</td>
<td>Specifies the slide numbers required to export when the <code>src</code> attribute points to a PowerPoint file. Use a hyphen to specify a range; use a comma to specify non-contiguous slides. For example: <code>slides=&quot;1-10&quot;</code> or <code>slides=&quot;1,10&quot;</code></td>
</tr>
</tbody>
</table>
| src       | Optional | HTML, SWF, or PPT source files used as a slide. You can specify the following as the slide source:  
  - An absolute path  
  - A path relative to the CFM page  
  - A URL: Specify if the source returns HTML content  
    SWF files must be present on the system running ColdFusion and the path must be either an absolute path or a path relative to the CFM page.  
    If you do not specify a source file, include HTML/CFML code as the body. If you specify a source file and HTML/CFML, ColdFusion ignores the source file and displays the HTML/CFML content in the slide. |
| title     | Optional | Title of the slide |
| userAgent | Optional | ColdFusion | Text to put in the HTTP User-Agent request header field. Identifies the request client software. |
### useExternalProgram

**Optional**

**true**

Boolean value to switch between OpenOffice and POI libraries:
- true: OpenOffice libraries.
- false: POI libraries

### video

**Optional**

Video file used for the presenter of the slide. If you specify video for the slide and an image for the presenter, the video is used instead of the image for the slide. You cannot specify both audio and video for a slide. The video must be an FLV or SWF file. The video file pathname must be relative to the CFM page or the web root.

---

**Usage**

Use the `cfpresentationslide` tag within the `cfpresentation` tag to create a slide presentation from individual SWF or HTML source files. If you do not specify a source file, include the HTML or CFML code for the body of the slide within the `cfpresentationslide` tag. You can assign one presenter to each slide. Use the `cfpresenter` tag to define presenters referenced by the `cfpresentationslide` tags.

The following code shows how to create a slide presentation from existing SWF files:

```cfml
<!--- The following example shows how to create a slide presentation --->
<!--- from individual SWF files located on the ColdFusion server. --->
<!--- Because no directory is specified, the presentation runs in --->
<!--- the browser. --->
<cfpresentation title="myPresentation">
  <cfpresentationslide title="1st slide" src="slide1.swf" duration="10"/>
  <cfpresentationslide title="2nd slide" src="slide2.swf"
    audio="audio1.mp3" duration="20"/>
  <cfpresentationslide title="3rd slide" src="slide3.swf"
    audio="audio2.mp3" duration="218"/>
</cfpresentation>
```

---

**Note**

The `cfpresentationslide` tag requires an end tag. If you specify a source file as the content for the slide instead of CFML and HTML code within start and end tags, use the end slash as a shortcut for the end tag.
You can reference source files from a URL as long as they return HTML content. The following code shows how to create a slide presentation from HTML files located on an external website:

```coldfusion
<!--- The following example shows how to create a slide presentation --->
<!--- from HTML files located on an external site. --->
<cfpresentation title="USGS Naming Conventions" directory="myPresentation">
  <cfpresenter name="Robert L. Payne" title="Executive Secretary">
    <cfpresenter name="Trent Palmer" title="Executive Secretary Foreign Names">
    </cfpresentationslide>
  </cfpresenter>
</cfpresentation>
```

**Note**

The links within slides created from HTML files are not active.

Also, you can enter HTML and CFML code as the body for a slide. Within the code, you can include charts, graphs, and images, as the following code shows:
<!--- This example shows how to create a slide presentation dynamically --->
<!--- from HTML code and CFML code. Because no directory is specified, --->
<!--- the presentation runs in the client browser. --->
<cfpresentation title="Sales Presentation">
    <cfpresenter name="Shyam" title="Vice President" email="shyam@somecompany.com">
        <cfpresenter name="Ram" title="Sr. Vice President" email="ram@somecompany.com">
            <cfpresentationslide title="Introduction" presenter="Shyam" audio="myAudio3.mp3"
                duration="10">
                <h3>Introduction</h3>
                <table>
                    <tr>
                        <td>
                            <ul>
                                <li>Overview</li>
                                <li>Q1 Sales Figures</li>
                                <li>Projected Sales</li>
                                <li>Competition</li>
                                <li>Advantages</li>
                                <li>Long Term Growth</li>
                            </ul>
                        </td>
                        <td><img src="../cfdocs/images/artgallery/paul01.jpg"/></td>
                    </tr>
                </table>
            </cfpresentationslide>
            <cfpresentationslide title="Q1 Sales Figures" duration="14" presenter="Ram"
                audio="myAudio1.mp3">
                <h3>Q1 Sales Figures</h3>
                <cfchart format="png" showborder="yes" chartheight="250" chartwidth="300"
                    pieslicestyle="sliced">
                    <cfchartseries type="pie">
                        <cfchartdata item="Europe" value="9"/>
                        <cfchartdata item="Asia" value="20"/>
                        <cfchartdata item="North America" value="50"/>
                        <cfchartdata item="South America" value="21"/>
                    </cfchartseries>
                </cfchart>
            </cfpresentationslide>
        </cfpresenter>
    </cfpresenter>
</cfpresentation>
<!--- The following example shows how to create a slide presentation --->
<!--- dynamically from HTML in ColdFusion and from HTML files located --->
<!--- on an external site. ColdFusion writes the presentation files --->
<!--- to a directory relative to the CFM page. --->
<cfpresentation title="USGS Naming Conventions" directory="namingConventions">
    <cfpresenter name="Robert L. Payne" title="Executive Secretary"/>
    <cfpresenter name="Trent Palmer" title="Executive Secretary Foreign Names"/>
    <cfpresentationslide presenter="Robert L. Payne">
        <h3>USGS Naming Conventions</h3>
        <ul>
            <li>Overview</li>
            <li>General Naming Conventions</li>
            <li>Domestic Naming Conventions</li>
            <li>Foreign Naming Conventions</li>
        </ul>
    </cfpresentationslide>
    <cfpresentationslide src="http://geonames.usgs.gov/domestic/index.html"
        duration="10" presenter="Robert L. Payne"/>
    <cfpresentationslide src="http://geonames.usgs.gov/foreign/index.html"
        duration="15" presenter="Trent Palmer"/>
</cfpresentation>
**cfpresenter**

**Description**

Describes a presenter in a slide presentation. A slide presentation can have multiple presenters. The presenters must be referenced from the slides defined by the `cfpresentationslide` tag.

**History**

ColdFusion 8: Added this tag.

**Category**

*Data output tags*

**Syntax**

```
<cfpresenter
    biography = "text string"
    email = "e-mail address of the presenter"
    image = "relative pathname for JPG"
    name = "text string"
    logo = "relative pathname for JPG"
    title = "text string">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

`cfchart`, `cfpresentation`, `cfpresentationslide`

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>biography</td>
<td>Optional</td>
<td></td>
<td>A text string that provides information about the presenter; for example, &quot;Sally Maverick has been a top seller of Adobe products for the last five years.&quot;</td>
</tr>
</tbody>
</table>
### cfpresenter Tag Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>email</strong></td>
<td>Optional</td>
<td>E-mail address of the presenter. This attribute activates the Contact link in the presentation control panel, which opens an e-mail message when you click it.</td>
</tr>
<tr>
<td><strong>image</strong></td>
<td>Optional</td>
<td>Pathname for the presenter's image in JPEG format. The JPEG file must be relative to the CFM page. If you specify a video for the <em>cfpresentationslide</em> tag, the video clip overrides this value for that slide.</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>Required</td>
<td>Name of the presenter. Use this value in the <em>presenter</em> attribute of the <em>cfpresentationslide</em> tag to associate the presenter with the slide.</td>
</tr>
<tr>
<td><strong>logo</strong></td>
<td>Optional</td>
<td>Pathname of the image file that represents the presenter's logo or the logo of the presenter's organization. The logo must be in JPEG format. The file must be relative to the CFM file website.</td>
</tr>
<tr>
<td><strong>title</strong></td>
<td>Optional</td>
<td>Title of the presenter, for example, &quot;VP of Sales&quot;.</td>
</tr>
</tbody>
</table>

### Usage

Use the `cfpresenter` tag to define the presenters that you specify for each slide. The presenter information appears in the control panel for the slide to which it is assigned. To specify a presenter for a slide, use the `presenter` attribute of the `cfpresentationslide` tag.

You can specify an image of the presenter and the presenter's company logo by using the `image` and `logo` attributes of the `presenter` tag, respectively. To display a video clip in place of the presenter's image, you can specify an FLV or SWF file for `video` attribute of the `cfpresentationslide` tag.

### Example
<!---- This example shows how to specify presenters for a slide --->
<!---- presentation and assign a presenter to each slide in the presentation. --->
<cfpresentation title="myPresentation" directory="presentation" overwrite="yes">

<!---- The following code defines three presenters. --->
<cfpresenter name="Shyam" title="President" email="shyam@somecompany.com"
  image="images\shyam01.jpg">
<cfpresenter name="Ram" title="V.P. Sales" email="ram@somecompany.com"
  image="images\ram01.jpg">
<cfpresenter name="Michelle" title="V.P. Engineering"
  email="mhatter@adobe.com" image="images\michelle01.jpg">

<!---- The following code assigns a presenter to each of three slides in the
presentation. --->
<cfpresentationslide title="myFirstSlide" src="slide1.swf" duration="10"
  presenter="Shyam"/>
<cfpresentationslide title="mySecondSlide" src="slide2.swf" duration="15"
  presenter="Michelle"/>
<cfpresentationslide title="myThirdSlide" src="slide3.swf" duration="2"
  presenter="Ram"/>

<!---- In the following slide, ColdFusion uses a video clip --->
<!---- instead of the JPEG image for the presenter. --->
<cfpresentationslide title="myFourthSlide" src="slide4.swf" duration="5"
  presenter="Shyam" video="video\video1.flv"/>
</cfpresentation>
**cfprint**

**Description**

Prints specified pages from a PDF file. Use this tag to perform automated batch print jobs. Use the `cfprint` tag to print any PDF document, including the ones generated by the `cfdocument`, `cfpdf`, and `cfpdfform` tag. Also, you use this tag to print Report Builder reports exported in PDF format.

**History**

ColdFusion 8: Added this tag.

**Category**

Data output tags

**Syntax**

```
<cfprint
    source = "absolute or relative pathname to a PDF file|PDF document variable"
    attributeStruct = "ColdFusion structure that contains standard print request key-value pairs"
    color = "yes|no"
    copies = "number of copies"
    fidelity = "yes|no"
    pages = "page or pages to print"
    password = "PDF source file owner or user password"
    paper = "letter|legal|A4|A5|B4|B5|B4-JIS|B5-JIS|any media supported by the printer"
    printer = "string that specifies the printer name"
    type = "PDF">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfdocument, cfpdf, cfpdfform, cfpdfformparam, cfpdfparam, cfpdfsubform, GetPrinterInfo, IsPDFFile, IsPDFObject

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>attributeStruct</th>
<th>Optional</th>
<th>ColdFusion structure used to specify additional print instructions. Individually named attributes take precedence over the key-value pairs in the attribute structure. For information about key-value pairs, see the table in the section attributeStruct.</th>
</tr>
</thead>
</table>
| color          | Optional | Color or monochrome printing:  
- **yes**: print in color  
- **no**: print in black and white, with colors in shades of gray |
| copies         | Optional | Number of copies to print. The value must be greater than or equal to 1. |
| fidelity       | Optional | no  
Whether to print a job based on the print requirements specified. Valid values are:  
- **yes**: if the job cannot be printed exactly as specified in the print requirements, the job is rejected.  
- **no**: a reasonable attempt to print the job is acceptable |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pages</td>
<td>Optional</td>
<td>all</td>
<td>Pages in the source file to print. Duplicate pages and pages beyond the total count of pages in the document are ignored as long as there is at least one page between 1 and the total number of pages in the document. You can combine individual page numbers and page ranges, for example, 1-3,6,10-20. If you do not specify a value for the <code>pages</code> attribute, ColdFusion prints the entire document.</td>
</tr>
</tbody>
</table>
| paper | Optional | | Paper used for the print job. The value can be any returned by the `GetPrinterInfo` function. The following values are valid:  
  - `na-letter`
  - `na-legal`
  - `iso-a4`
  - `iso-a5`
  - `iso-b4`
  - `iso-b5`
  - `jis-b4`
  - `jis-b5`
  For more information, see the section Supported paper types. |
| password | Optional | | The owner or user password for the PDF source file. If the PDF file is password-protected, specify this attribute for the file to print. |
### printer
Optional
The name of a printer. An example in Windows is `\s1001prn02\NTN-2W-HP_BW02`. The default name is the default printer for the account where the ColdFusion server is running. Printer names are case sensitive and must be entered exactly as they appear in the System Information page of the ColdFusion Administrator. For more information, see Usage.

### source
Required
Source document to print. Specify one of the following:
- An absolute or relative pathname to an on-disk or in-memory PDF document file, for example, `c:\work\myPDF.pdf` or `myPDF.pdf`. The default directory is the template directory.
- A PDF document variable in memory that is generated by the `cfdocument` tag or the `cfpdf` tag, for example, "myPDFdoc".

### type
Optional
PDF
The file type of the document being printed. The only valid value is PDF.

### Usage
Use the `cfprint` tag for automated batch printing of PDF documents. For example, you can run a batch job each evening that generates a report in PDF format and then prints either the entire report or selected pages for review the next morning without user intervention. Most of the `cfprint` tag attributes are printer-dependent. If a printer does not support a specified attribute, it ignores the instruction. The default settings for the attributes also are printer-dependent. If you set a default printer, only specify the PDF file source and the password, if the file is password-protected.
Note

The particular printer attributes supported are dependent on the operating system, the network printer server, if there is one, and the printer. The `cfprint` tag is dependent on the Java Print Service (JPS). Many printers support attributes that are not accessible from JPS. For example, the JPS for a Macintosh OSX running JDK 1.5 supports the fewest printer attributes. Upgrading to JDK 1.6 adds some functionality, but finishing attributes are still not supported.

If the `fidelity` attribute is set to `yes`, the job does not print if any of the specified attributes are not supported by the printer. If the `fidelity` attribute is set to `no`, the printer accepts the print job and either ignores any attribute it does not support or substitutes a reasonable alternative for the attribute.

To determine which attributes are supported on a specified printer, use the `GetPrinterInfo` function.

Supported paper types

You can use the equivalent page types supported by the `cfdocument` tag, but they are not returned by the `GetPrinterInfo` function:

<table>
<thead>
<tr>
<th>cfdocument</th>
<th>cfprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>• letter</td>
<td>• na-letter</td>
</tr>
<tr>
<td>• legal</td>
<td>• na-legal</td>
</tr>
<tr>
<td>• A4</td>
<td>• iso-a4</td>
</tr>
<tr>
<td>• A5</td>
<td>• iso-a5</td>
</tr>
<tr>
<td>• B4</td>
<td>• iso-b4</td>
</tr>
<tr>
<td>• B5</td>
<td>• iso-b5</td>
</tr>
<tr>
<td>• B4-JIS</td>
<td>• jis-b4</td>
</tr>
<tr>
<td>• B5-JIS</td>
<td>• jis-b5</td>
</tr>
</tbody>
</table>

View a list of configured printers

1. Log on to the ColdFusion Administrator.
2. Click the System Information icon located at the top right of the Administrator Console window. (The icon has an "i" on it.)
3. Scroll to the bottom of the System Information page. Under Printer Details is the Default Printer. Below the default printer is Printers, which lists the configured printers available to ColdFusion, including the default printer.

Printer configuration is operating system-dependent. Configure printers outside of ColdFusion.

View the print log

1. Log on to the ColdFusion Administrator.
2. Expand the Debugging and Logging topic.
3. Click the Log Files link. The print.log file appears in the list of log files.

Permissions for printing

If the PDF file is encrypted, the permissions for the file must be set to `AllowPrinting`, or specify the owner password to print the file. Use the `protect` action of the `cfpdf` tag to set permissions and passwords on PDF files. For more information, see Permissions for PDF documents in `cfpdf`.

If a Security Manager is installed, the following permission is required in the coldfusion.policy file to initiate a print job request:
grant { permission java.lang.RuntimePermission "queuePrintJob"; };

In Windows systems, the account running the ColdFusion server must have PRINTER_ACCESS_USE access rights for each printer it uses. Even if the printer is configured locally on the system, the printer is not available if the account in which ColdFusion is running does not have the proper permissions.

⚠️ Note

By default, ColdFusion installs and runs as the Local System account, which may not have printer queue access rights. For information on running ColdFusion as a specific user, see the following Tech Note: [http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=tn_17279](http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=tn_17279)

**attributeStruct**

The following table lists the optional attributeStruct key-value pairs that you use to specify print requests:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoRotateAndCenter</td>
<td>Adjusts the document's orientation to match the orientation specified in the printer attributes and centers the page in the imaging area:</td>
</tr>
<tr>
<td></td>
<td>• yes: the orientation, if specified, is ignored (default)</td>
</tr>
<tr>
<td></td>
<td>• no: the orientation, if specified, is applied to the document</td>
</tr>
<tr>
<td>collate or sheetCollate</td>
<td>Specifies whether the sheets of each copy of each printed document in a job are in sequence when multiple copies of the document are specified by the copies attribute:</td>
</tr>
<tr>
<td></td>
<td>• yes</td>
</tr>
<tr>
<td></td>
<td>• no</td>
</tr>
<tr>
<td>color or chromaticity</td>
<td>Specifies color or monochrome printing. Monochrome printing displays colors in shades of gray:</td>
</tr>
<tr>
<td></td>
<td>• yes: print in color.</td>
</tr>
<tr>
<td></td>
<td>• no: print in monochrome.</td>
</tr>
<tr>
<td>copies</td>
<td>Number of copies of the source document to print. Valid values are integers greater than or equal to 1.</td>
</tr>
<tr>
<td>coverPage or jobSheets</td>
<td>Specifies which job start and end sheets, if any, are printed with a job:</td>
</tr>
<tr>
<td></td>
<td>• none</td>
</tr>
<tr>
<td></td>
<td>• standard</td>
</tr>
</tbody>
</table>
| **fidelity** | Specifies whether to print a job based on the print requirements specified. The following values are valid values:  
  - **yes**: If the job cannot be printed exactly as specified in the print requirements, the job is rejected.  
  - **no**: A reasonable attempt to print the job is acceptable (default). |
| **finishings** | Finishing operation to perform after each copy of a document is printed:  
  - staple-top-left  
  - staple-bottom-left  
  - staple-top-right  
  - staple-bottom-right  
  - edge-stitch-left  
  - edge-stitch-right  
  - edge-stitch-top  
  - edge-stitch-bottom  
  - dual-right  
  - dual-top  
  - dual-bottom  
  - dual-left |
| **jobHoldUntil** | Date-time attribute for the exact date and time at which the job is available for printing. Valid values are ColdFusion date and time variables. |
| **jobName** | The name of a print job. |
| **jobPriority** | Integer value that represents a print job's priority. Among those jobs that are ready to print, a printer must print all jobs with a priority value of \( n \) before printing those with a priority value of \( n-1 \) for all \( n \). Valid values are integers from 1 (lowest priority) through 100 (highest priority). |
| **numberUp** | Number of pages to print on a single side of paper. The value must be a number greater than or equal to 1. |
| **orientation or orientationRequested** | Orientation of the page to be printed. The only valid value for PDF documents is **portrait**. To change the orientation to landscape, set the `autoRotateAndCenter` to **yes** (which is the default value). The `autoRotateAndCenter` instruction overrides the **orientation** instruction. |
| pages | Pages in the source file to print. Duplicate pages and pages beyond the total count of pages in the document are ignored as long as there is at least one page between 1 and the total number of pages in the document. You can combine individual page numbers and page ranges, for example, 1-3,6,10-20. If you do not specify a value for the `pages` attribute, ColdFusion prints the entire document. |
| pageScaling | Specifies how pages are scaled on the paper:  
- **fit-to-printer-margins**: Reduces or enlarges each page to fit the printable area of the currently selected paper size.  
- **reduce-to-printer-margins**: Shrinks large pages to fit the currently selected paper size but does not enlarge small pages. If an area is selected and is larger than the printable area of the currently selected paper, the page is scaled to fit the printable area (Default).  
- **none**: Prints the upper left or center of a page (if autorotated and centered) without scaling. Pages that don't fit on the paper are cropped. |
| pageSubset | Prints a subset of pages in specified by the `pages` attribute:  
- **all**: Prints all the pages in the specified page range (Default).  
- **odd**: Prints only the odd pages in the specified page range.  
- **even**: Prints only the even pages in the specified page range. |
| paper | Paper used for the print job. The value can be any returned by the `GetPrinterInfo` function. The following values are the most common:  
- **na-letter**  
- **iso-a4** |
| presentationDirection | Used in conjunction with the `numberUp` attribute to indicate the layout of multiple document pages on one side of the paper. |
### printer

The name of a printer. An example in Windows is `\s1001prn02\NTN-2W-HP_BW02`. The default name is the default printer for the account where the ColdFusion server is running. Printer names are case sensitive and you must enter the names exactly as they appear in the System Information page of the ColdFusion Administrator. For more information on viewing print logs, see Usage.

### quality

Print quality for the print job:
- draft
- high
- normal

### requestingUserName

A string that specifies the name of the end user that submitted the print job.

### reversePages

Prints pages in reverse order. If page ranges are entered, the pages print opposite of the order in which they were entered. For example, if the Pages box shows 3-5, 7-10, selecting Reverse Pages prints pages 10-7, and then 5-3.
- yes
- no (default)

### sides

Sides of the paper on which the pages are printed:
- one-sided (default)
- duplex or two-sided-long-edge
- tumble or two-sided-short-edge

### usePdfPageSize

Uses the PDF page size to determine the area of the paper printed rather than the paper size. This is useful for printing PDF documents that contain multiple page sizes:
- yes
- no (default)

### Example

The following example shows how to use the `attributeStruct` attribute and the `cfprint` tag to print five, double-sided copies of a letter-sized PDF document, which are stapled on the top-left corner and collated:

```cfml
<cfset aset=StructNew()>  
<cfset aset["sides"] = "duplex">  
<cfprint type="pdf" source="myfile.pdf" 
    printer="\s1001prn02\NTN-2W-HP_BW02" copies="5" paper="letter" 
    attributeStruct="#aset#">  
```
The following example shows how to specify all of the print attributes with the `attributeStruct` attribute:

```cfml
<cfset aset=StructNew()>
<cfset aset["paper"] = "letter">
<cfset aset["sides"] = "duplex">
<cfset aset["copies"] = "5">
<cfset aset["printer"] = "\s1001prn02\NTN-2W-HP_BW02">

<cfprint type="pdf" source="myfile.pdf" attributeStruct="#aset#">
```

Printers have a setting called `autoRotateAndCenter`, which is set to `yes` by default. The following example shows how to override the default `autoRotateAndCenter` setting and use the `orientation` setting instead:

```cfml
<cfset aset=StructNew()>
<cfset aset["autoRotateAndCenter"] = "no">
<cfset aset["orientation"] = "portrait">

<cfprint printer="myprinter" source="_mydoc.pdf" attributeStruct="#aset#">
```

To run a print job asynchronously, start a print job in a thread. Do not wait for the print job to be sent to the printer before proceeding. To start a print job in a thread, enclose the `cfprint` tag within `cfthread` start and end tags, as the following example shows:

```cfml
<cfthread name="mythread" action="run">
  <cfprint type="pdf" source="myfile.pdf" printer="\s1001prn02\NTN-2W-HP_BW02">
</cfthread>
```

For more information, see `cfthread`. 
cfprocessingdirective

Description

Provides the following information to ColdFusion about how to process the current page:

- Specifies whether to remove excess whitespace character from ColdFusion generated content in the tag body.
- Identifies the character encoding (character set) of the page contents.

Category

Data output tags, Page processing tags

Syntax

```xml
<cfprocessingdirective
    pageencoding = "page-encoding literal string">

OR

<cfprocessingdirective
    pageEncoding = "page-encoding literal string"
    suppressWhiteSpace = "yes|no">
    CFML tags
</cfprocessingdirective>
```

See also

cfcol, cfcontent, cfoutput, cfsetting, cfsilent, cftable; Developing Globalized Applications in the Developing ColdFusion Applications

History

ColdFusion MX:

- Changed suppresswhitespace attribute value behavior: you can specify the suppresswhitespace attribute value as a string variable. (ColdFusion 5 supported setting it only as a constant.)
- Added the pageEncoding attribute.

ColdFusion 11:

- In general, one should no longer need to use this tag to specify the page encoding as the ColdFusion server should be able to identify it automatically.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>attribute</th>
<th>type</th>
<th>description</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageEncoding</td>
<td>Optional</td>
<td>Character encoding identified by the page byte order mark, if any; otherwise, system default encoding</td>
<td>A string literal; cannot be a variable. Identifies the character encoding of the current CFML page. This attribute affects the entire page, not just the cfprocessingdirective tag body. The value may be enclosed in single- or double-quotation marks, or none. The following list includes commonly used values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• utf-8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• iso-8859-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• windows-1252</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• us-ascii</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• shift_jis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• iso-2022-jp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• euc-jp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• euc-kr</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• big5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• euc-cn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• utf-16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information on character encodings, see <a href="http://www.w3.org/International/O-charset.html">www.w3.org/International/O-charset.html</a>.</td>
<td></td>
</tr>
<tr>
<td>suppressWhiteSpace</td>
<td>Optional</td>
<td>Boolean; whether to suppress white space characters within the cfprocessingdirective block that are generated by CFML tags and often do not affect HTML appearance. Does not affect any white space in HTML code.</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

The cfprocessingdirective tag has limitations that depend on the attribute you use. For this reason, Adobe recommends that you include either the pageEncoding or suppressWhiteSpace attribute in a cfprocessingdirective tag, not both. To specify both values, use separate tags.

In a ColdFusion component (.cfc file), the cfprocessingdirective tag must follow the cfcomponent tag. If you use the pageEncoding attribute, the following rules apply:

- You must put the tag within the first 4096 bytes of a page. It can be positioned after a cfsetting or cfinclude tag.
- If you use the tag on a page that includes other pages by using the cfinclude or cfmodule tags, custom
tag invocation, and so on, the tag has no effect on the included pages.

- You cannot embed the tag within conditional logic, because the pageEncoding attribute is evaluated when ColdFusion compiles a page (not when it executes the page). For example, the following code has no effect at execution time, because the cfprocessingdirective tag has already been evaluated:

  `<cf if dynEncoding is not "dynamic encoding is not possible">
    <cfprocessingdirective pageencoding=#dynEncoding#>
  </cf if>`

- If you have multiple cfprocessingdirective tags in one page that specify the pageEncoding attribute, they must all specify the same value; if not, ColdFusion throws an error.
- If you specify only the pageencoding attribute, do not use a separate end tag.
- ColdFusion accepts character encoding names that are supported by the Java platform. If an invalid name is specified, ColdFusion throws an InvalidEncodingSpecification exception.
- If a page has a byte order mark (BOM), and a pageencoding attribute specifies an encoding that differs from the BOM, ColdFusion generates an error.

The following rules apply to the suppressWhiteSpace attribute:

- You can specify the suppresswhitespace attribute value as a constant or a variable. To use a variable: define the variable (for example, whitespaceSetting), assign it the value yes or no, and code a statement such as the following:

  <!--- ColdFusion allows suppression option to be set at runtime --->
  <cfprocessingdirective suppresswhitespace=#whitespaceSetting#>  
  code to whose output the setting is applied 
  </cfprocessingdirective>

- The suppresswhitespace attribute only affects code that you put between the <cfprocessingdirective> begin tag and the </cfprocessingdirective> end tag.

The following example shows the use of a nested cfprocessingdirective tag. The outer tag suppresses unnecessary whitespace during computation of a large table; the inner tag retains whitespace, to output a preformatted table.

Example

```cfprocessingdirective suppressWhiteSpace = "Yes">
  <!--- CFML code --->
  <cfprocessingdirective suppressWhiteSpace = "No">
    <cfoutput>#table_data#</cfoutput>
  </cfprocessingdirective>
</cfprocessingdirective>
```

The following example shows the use of the pageencoding attribute:

```cfprocessingdirective pageencoding = "shift_jis"```
### cfprocparam

**Description**

Defines stored procedure parameters. This tag is nested within a `cfstoredproc` tag.

**Category**

*Database manipulation tags*

**Syntax**

```xml
<cfprocparam
    CFSQLType = "parameter data type"
    maxLength = "length"
    null = "yes|no"
    scale = "decimal places"
    type = "in|out|inout"
    value = "parameter value"
    variable = "variable name">
</cfprocparam>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfinsert, cfprocreuslt, cfquery, cfqueryparam, cfstoredproc, cftransaction, cfupdate; Optimizing ColdFusion applications in [Designing and Optimizing a ColdFusion Application](https://coldfusion.adobe.com) in the *Developing ColdFusion Applications*

**History**

ColdFusion MX:

- The `maxrows` attribute is obsolete.
- Changed the `dbvarname` attribute behavior: it is now ignored for all drivers. ColdFusion uses JDBC 2.2 and does not support named parameters. This is deprecated.
- Changed the `maxLength` attribute behavior: it now applies to IN and INOUT parameter values.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFSQLType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxLength</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>null</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scale</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>variable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFSQLType</td>
<td>Required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SQL type to which the parameter (any type) is bound. ColdFusion supports the following values, where the last element of the name corresponds to the SQL data type. Different database systems might support different subsets of this list. For information on supported parameter types, see your DBMS documentation.

- CF_SQL_BIGINT
- CF_SQL_BIT
- CF_SQL_BLOB
- CF_SQL_CHAR
- CF_SQL_CLOB
- CF_SQL_DATE
- CF_SQL_DECIMAL
- CF_SQL_DOUBLE
- CF_SQL_FLOAT
- CF_SQL_IDSTAMP
- CF_SQL_INTEGER
- CF_SQL_LONGVARCHAR
- CF_SQL_MONEY
- CF_SQL_MONEY4
- CF_SQL_NUMERIC
- CF_SQL_REAL
- CF_SQL_REFCURSOR
- CF_SQL_SMALLINT
- CF_SQL_TIME
- CF_SQL_TIMESTAMP
- CF_SQL_TINYINT
- CF_SQL_VARCHAR

For a mapping of ColdFusion SQL data types to JDBC data types, see [cfqueryparam](https://help.coldfusion.com/adobe-coldfusion-documentation/).
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxLength</td>
<td>Maximum length of a string or character IN or INOUT value attribute. A maxLength of 0 allows any length. The maxLength attribute is not required when specifying type=out.</td>
</tr>
</tbody>
</table>
| null        | Whether the parameter is passed in as a null value. Not used with OUT type parameters.  
  - yes: tag ignores the value attribute.  
  - no                                           |
| scale       | Number of decimal places in numeric parameter. A scale of 0 limits the value to an integer. |
| type        |  
  - in: the parameter is used to send data to the database system only. Passes the parameter by value.  
  - out: the parameter is used to receive data from the database system only. Passes the parameter as a bound variable.  
  - inout: the parameter is used to send and receive data. Passes the parameter as a bound variable. |
| value       | Value that ColdFusion passes to the stored procedure. This is optional for inout parameters.  
  Required if type = "IN"
**variable**

Required if `type = "OUT"` or "INOUT"

ColdFusion variable name; references the value that the output parameter has after the stored procedure is called. This is ignored for *in parameters.*

---

**Usage**

Use this tag to identify stored procedure parameters and their data types. Code one `cfprocparam` tag for each parameter. The parameters that you code vary based on parameter type and DBMS. ColdFusion supports positional parameters. If you use positional parameters, you must code `cfprocparam` tags in the same order as the associated parameters in the stored procedure definition. Output variables are stored in the ColdFusion variable specified by the `variable` attribute.

You cannot use the `cfprocparam` tag for Oracle 8 and 9 reference cursors. Instead, use the `cfprocresult` tag.

---

**Example**

The following examples list the equivalent Oracle and Microsoft SQL Server stored procedures that insert data into the database. The CFML to invoke either stored procedure is the same. The following example shows the Oracle stored procedure:

```sql
CREATE OR REPLACE PROCEDURE Insert_Book (  
    arg_Title Books.Title%type,  
    arg_Price Books.Price%type,  
    arg_PublishDate Books.PublishDate%type,  
    arg_BookID OUT Books.BookID%type)  
AS  
    num_BookID NUMBER;  
BEGIN  
    SELECT seq_Books.NEXTVAL  
    INTO num_BookID  
    FROM DUAL;  

    INSERT INTO  
    Books (  
        BookID,  
        Title,  
        Price,  
        PublishDate )  
    VALUES (  
        num_BookID,  
        arg_Title,  
        arg_Price,  
        arg_PublishDate );  

    arg_BookID := num_BookID;  
END;  
/```

The following example shows the SQL Server stored procedure:
CREATE PROCEDURE Insert_Book (  
  @arg_Title VARCHAR(255),
  @arg_Price SMALLMONEY,
  @arg_PublishDate DATETIME,
  @arg_BookID INT OUT)
AS
BEGIN
INSERT INTO
Books (  
  Title,
  Price,
  PublishDate )
VALUES (  
  @arg_Title,
  @arg_Price,
  @arg_PublishDate );

SELECT @arg_BookID = @@IDENTITY;
END;

You use the following CFML code to call either stored procedure:
<cfset ds = "sqltst">
<!-- <cfset ds = "oratst" -->

<!--- If submitting a new book, insert the record and display confirmation --->
<cfif isDefined("form.title")>
<cfstoredproc procedure="Insert_Book" datasource="#ds#">
<cfprocparam cfsqltype="cf_sql_varchar" value="#form.title#">
<cfprocparam cfsqltype="cf_sql_numeric" value="#form.price#">
<cfprocparam cfsqltype="cf_sql_date" value="#form.price#">
<cfprocparam cfsqltype="cf_sql_numeric" type="out" variable="bookId">
</cfstoredproc>

<cfoutput>
<h3>'#form.title#' inserted into database. The ID is #bookId#.</h3>
</cfoutput>
</cfif>

<cfform action="#CGI.SCRIPT_NAME#" method="post">
<h3>Insert a new book</h3>

Title:  
<cfinput type="text" size="20" required="yes" name="title"/>
<br/>

Price:  
<cfinput type="text" size="20" required="yes" name="price" validate="float"/>
<br/>

Publish Date:  
<cfinput type="text" size="5" required="yes" name="publishDate" validate="date"/>
<br/>

<input type="submit" value="Insert Book"/>

</cfform>
**cfprocresult**

**Description**

Associates a query object with a result set returned by a stored procedure. Other ColdFusion tags, such as `cfoutput` and `cftable`, use this query object to access the result set. This tag is nested within a `cfstoredproc` tag.

**Category**

*Database manipulation tags*

**Syntax**

```
<cfprocresult
  name = "query name"
  maxRows = "number"
  resultSet = "1-n">
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

`cfinsert`, `cfprocparam`, `cfquery`, `cfqueryparam`, `cfstoredproc`, `cftable`, `cftransaction`, `cfupdate`; *Optimizing database use* in *Designing and Optimizing a ColdFusion Application* in the *Developing ColdFusion Applications*

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Name for the query result set.</td>
</tr>
<tr>
<td>maxRows</td>
<td>Optional</td>
<td>-1 (All)</td>
<td>Maximum number of rows returned in result set.</td>
</tr>
<tr>
<td>resultSet</td>
<td>Optional</td>
<td>1</td>
<td>Names one result set, if stored procedure returns more than one.</td>
</tr>
</tbody>
</table>

**Usage**

To enable access to data returned by the stored procedure, specify one or more `cfprocresult` tags. If the stored procedure returns more than one result set, use the `resultSet` attribute to specify which of the stored procedure's result sets to return.

The `resultSet` attribute must be unique within the scope of the `cfstoredproc` tag. If you specify a result set twice, the second occurrence overwrites the first. CFML supports Oracle 8 and 9 Reference Cursor type, which passes a parameter by reference. Parameters that are passed this way can be allocated and deallocated from
memory within the execution of one application. To use reference cursors in packages or stored procedures, use the `cfprocresult` tag. This causes the ColdFusion JDBC database driver to put Oracle reference cursors into a result set. (You cannot use this method with Oracle’s ThinClient JDBC drivers.)

Example

```cfml
<!--- This example executes a Sybase stored procedure that returns three result sets, two
of which we want. The stored procedure returns status code and one output parameter, which
we display. We use named notation for parameters. --->
<!---- cfstoredproc tag ---->
<cfstoredproc procedure = "foo_proc"
    dataSource = "MY_SYBASE_TEST" username = "sa"
    password = "" dbServer = "scup" dbName = "pubs2"
    returnCode = "Yes" debug = "Yes">

<!--- cfprocresult tags --->
<cfprocresult name = RS1>
<cfprocresult name = RS3 resultSet = 3>

<!--- cfprocparam tags --->
<cfprocparam type = "IN" CFSQLType = CF_SQL_INTEGER value = "1">
<cfprocparam type = "OUT" CFSQLType = CF_SQL_DATE variable = FOO>

<!--- Close the cfstoredproc tag. --->
</cfstoredproc>
<cfoutput>
The output param value: '#foo#'<br></cfoutput>

<h3>The Results Information</h3>
<cfoutput query = RS1>#name#, #DATE_COL#<br></cfoutput>
<p></p>
<cfoutput query = RS3>#col1#, #col2#, #col3#<br></cfoutput>
<p></p>
<p>Record Count: #RS1.recordCount# Columns: #RS1.columnList#</p>
<p>Record Count: #RS3.recordCount# Columns: #RS3.columnList#</p>
The return code for the stored procedure is: '#cfstoredproc.statusCode#'<br></cfoutput>
...
cfprogressbar

Description

Defines a progress bar to indicate the progress of an activity such as a file download.

Category

Display management tags

Syntax

```
<cfprogressbar
    autoDisplay="true|false"
    name="control identifier"
    bind ="bind expression"
    duration="time value"
    height="height in pixels"
    interval="time in milliseconds"
    onComplete="function name"
    onError="JavaScript function name"
    style="style specification"
    width="pixel value"
>
```

History

ColdFusion 9: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoDisplay</td>
<td>Optional</td>
<td>true</td>
<td>Set to true to display the progress bar.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>The control name. Used to refer to the control in JavaScript, for example in the script that starts the progress.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Required if</td>
<td>Examples</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>bind</td>
<td>Required if duration is not specified</td>
<td>A bind expression specifying a client JavaScript function or server CFC that the control calls to get progress information each time the period defined by the interval attribute elapses. You cannot use this attribute with a duration attribute. If an error occurs, no further bind expression calls are triggered and on Error is called (if defined).</td>
<td></td>
</tr>
<tr>
<td>duration</td>
<td>Required if bind is not specified</td>
<td>The time, in milliseconds, between when the bar starts showing progress and when it shows completed progress. Use only on automatic progress bars that do not use a bind expression to get actual progress information. You cannot use this attribute with a bind attribute.</td>
<td></td>
</tr>
<tr>
<td>height</td>
<td>Optional</td>
<td>Height of the bar in pixels.</td>
<td></td>
</tr>
<tr>
<td>interval</td>
<td>Optional</td>
<td>1000</td>
<td>Used if duration is specified. The time interval, in milliseconds, at which the progress bar updates. Although this attribute is optional, specify it to prevent the bar from updating frequently.</td>
</tr>
<tr>
<td>onComplete</td>
<td>Optional</td>
<td>The name of a function to call when progress completes.</td>
<td></td>
</tr>
</tbody>
</table>
onError | Applies only if you use bind. The JavaScript function to run on an error condition. The error can be a network error or server-side error.

style | Optional | The following are the supported colors:

- bgcolor: The background color for the progress bar. A hexadecimal value without "#" prefixed.
- textcolor: Text color on progress bar.
- progresscolor: Color used to indicate the progress.

width | Optional | 400 | The width (length) of the bar in pixels.

Usage

A progress bar has one of two behaviors:

- Manual, where the progress indicator length increases steadily over a time specified by the duration attribute.
- Dynamic, where the bind attribute specifies a function that determines the indicator length.

If you use a bind expression, the called function takes no parameters, and must return a structure with two values:

- status - A decimal completion value, in the range 0 - 1.0
- message - A message to display in the progress bar, such as "Loading..." or "Completed".

You use two Ajax functions to start and stop the progress bar:

```
ColdFusion.ProgressBar.start(barName)
ColdFusion.ProgressBar.stop(barName)
```

You must call the `start` method to start the progress bar. You call the `stop` method to explicitly stop the progress bar. The bar stops automatically when the bind method returns a `status` value of 1 or the period specified by the `duration` attribute elapses. Therefore, you need to use this method only if a process does not complete, if the process completes before the `duration` period, or in other nonstandard situations, such as error conditions.

Example

The following uses a simple comment form, and uses a timer to simulate the time it would take to process the form. The Application.cfc page must enable session management.
The main application page contains the following code:
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
</head>

<script type="text/javascript">
function startProgress() {
  ColdFusion.ProgressBar.start("mydataProgressbar");
}

function onfinish() {
  alert("Done");
}
</script>

<body>
<!---- Ensure there is no Session.STATUS value, which is used by the progress bar bind CFC, when the page displays. ---->
<cfif IsDefined("Session.STATUS")>
<cfscript>
  StructDelete(Session,"STATUS");
</cfscript>
</cfif>

<!---- For code simplicity, formatting is minimal. ---->
<cfform name="kitform">
  <p>To make our service better and to benefit from our special offers, take a moment to give us your email address and send us a comment.</p>
  <p>Name:<br />
    &nbsp;<cfinput type="text" name="name">  </p>
  <p>E-mail:<br />
    &nbsp;<cfinput type="text" name="email">  </p>
  <p>Comment:<br />
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    <cfinput type="button" name="" value="Send Comment" onclick=startProgress()></p>
</cfform>

<!--- The progressbar control --->
<div style="padding-left:3px">
  <cfprogressbar name="mydataProgressbar"
    bind="cfc:mycfc.getstatus()"
    interval="1700"
    width="200"
    oncomplete="onfinish"/>
</div>

</body>
</html>
The mycfc.cfc file has a single `getstatus` function:

```xml
<cfcomponent>
<!--- This function simulates the time taken by and operation by using sleep functions. It increments the progressbar status by 1/10 each time the progressbar bind expression calls it (that is, each time the time specified by the cfprogressbar interval attribute passes. --->
<cffunction name="getstatus" access="remote">
   <cfset str = StructNew()>
   <cfset str.message = "Saving Data">
   <cfif NOT IsDefined("session.STATUS")>
      <cfset session.STATUS = 0.1>
      <cfscript>
         Sleep(200);
      </cfscript>
      <cfelseif session.STATUS LT 0.9>
         <cfset session.STATUS = session.STATUS + .1>
         <cfscript>
            Sleep(200);
         </cfscript>
      </cfelse>
      <cfelse>
         <cfset str.message = "Done...">
         <cfset session.STATUS = "1.0">
      </cfif>
   </cfif>
   <cfreturn str>
</cffunction>
</cfcomponent>
```
**cfproperty**

**Description**

Defines properties and their annotations for a ColdFusion component (CFC). The properties are used to create complex data types for web services, while the annotations are used to define Object Relational Model (ORM) for a CFC. The attributes of this tag are exposed as component metadata and are subject to inheritance rules.

**Category**

*Extensibility tags*

**Syntax**

```xml
<cfproperty
    name="name"
    default="default value"
    displayname="descriptive name"
    hint="extended description"
    required="false|true"
    serializable="true|false"
    type="type">
```

**Note**

For ORM-related attributes and their usage, see *Map the properties* in the Developing ColdFusion Applications.

**See also**

*cfargument, cffunction, cfinvoke, cfinvokeargument, cfoject, cfreturn*; Documenting CFCs in *Building and Using ColdFusion Components*, *Implicit Get and Set Functions* in the Developing ColdFusion Applications

**History**

ColdFusion 9: Added attributes for defining Object Relational Model for the CFC. Added implicit getters and setters. Added validate and validateparams attributes.

ColdFusion MX: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>batchsize</td>
<td></td>
<td></td>
<td>For information about these attributes, see <em>Cold Fusion ORM</em>.</td>
</tr>
<tr>
<td>cascade</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>catalog</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>collectiontype</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>column</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constrained</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>datatype</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>Optional if no property value is set when the component is used for a web service, specifies a default value. If this attribute is present, the <code>required</code> attribute must be set to <code>no</code> or not specified. For ORM-specific usage of the default attribute, see <a href="https://help.adobe.com">ColdFusion ORM</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>Optional A value to be displayed when using introspection to show information about the CFC. The value appears in parentheses following the property name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dynamicInsert</td>
<td>For information about these attributes, see <a href="https://help.adobe.com">ColdFusion ORM</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dynamicupdate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elementColumn</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elementType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>entityname</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fetchbatchsize</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fieldType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fkcolumn</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>formula</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>generator</td>
<td>Optional</td>
<td>Specifies whether to generate getter methods or not. Value are: true, false</td>
<td></td>
</tr>
<tr>
<td>getter</td>
<td>Optional</td>
<td>Text to be displayed when using introspection to show information about the CFC. This attribute can be useful for describing the purpose of the parameter.</td>
<td></td>
</tr>
<tr>
<td>hint</td>
<td>Optional</td>
<td>For information about these attributes, see Cold Fusion ORM.</td>
<td></td>
</tr>
<tr>
<td>index</td>
<td></td>
<td>For information about these attributes, see Cold Fusion ORM.</td>
<td></td>
</tr>
<tr>
<td>insert</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inverse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inversejoin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>joincolumn</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lazy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>length</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linkcatalog</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linkschema</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linktable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mappedby</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>missingrowIgnored</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td>A string; a property name. Must be a static value.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Value</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>notnull</td>
<td>For information about these attributes, see Cold Fusion ORM.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>optimisticLock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>optimisticLockgenerated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>orderby</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>orderByreadonly</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>params</td>
<td>For information about these attributes, see Cold Fusion ORM.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>persistent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>preceision</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>readonly</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>readonly</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>required</td>
<td>Optional</td>
<td>no</td>
<td>Whether the parameter is required:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no</td>
</tr>
<tr>
<td>rowid</td>
<td>For information about these attributes, see Cold Fusion ORM.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scale</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>setter</td>
<td>Specifies whether to generate setter methods or not. Possible values are:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• false</td>
</tr>
<tr>
<td>schema</td>
<td>For information about these attributes, see Cold Fusion ORM.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Serializable

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>selectbeforeupdate</code></td>
<td>Optional</td>
<td>true</td>
<td>Specifies whether this property can be serialized. If you set this value to <code>false</code>, the property cannot be serialized, so any changes made are not retained on session replication, and the property has its default value (if any) on the second server. Use this attribute to prevent serialization of properties in CFCs that are serializable.</td>
</tr>
<tr>
<td><code>selectkey</code></td>
<td></td>
<td></td>
<td>For information about these attributes, see <a href="https://help.adobe.com/">Cold Fusion ORM</a>.</td>
</tr>
<tr>
<td><code>sequence</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>table</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>structkeycolumn</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>structkeydatatype</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>structkeyType</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>table</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
<td>any</td>
<td>A string; identifies the property data type:</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>-----</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• any</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• array</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• binary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• guid: the argument must be a UUID or GUID of the form xxxxxxxx-xxxx-xxxx-xxx x-xxxxxxxxxxxx where each x is a character representing a hexadecimal number (0-9A-F).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• numeric</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• query</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• struct</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• uuid: The argument must be a ColdFusion UUID of the form xxxxxxxx-xxxx-xxxx-xxx xxxxxxxxxxxxxxx where each x is a character representing a hexadecimal number (0-9A-F).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• variableName: a string formatted according to ColdFusion variable naming conventions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• a component name: if the type attribute value is not one of the preceding items, ColdFusion treats it as the name of a ColdFusion component. When the function executes, it generates an error if the argument that is passed in is not a CFC with the specified name.</td>
</tr>
</tbody>
</table>

| unique   |          |     | For information about these attributes, see [Cold Fusion ORM](http://example.com). |
validate Optional
validateparam Optional

Usage

You must position `<cfproperty>` tags at the beginning of a component, above executable code and function definitions. If a component is not used as a web service, `<cfproperty>` only provides metadata information of the property. It does not define variables or set values that you can use in your component. However, it creates implicit setters and getters for the property in the CFC depending on whether getter/setter attributes are enabled. For details, see Implicit Get and Set Functions in Developing ColdFusion Applications.

For Object Relational Model (ORM), `<cfproperty>` is used to define relational mapping for the property of the CFC. For details, see ColdFusion ORM in Developing ColdFusion Applications.

For web services that you create in ColdFusion, the `<cfproperty>` tag defines complex variables used by the web service.

Example

The following code defines a component in the file address.cfc that contains properties that represent a street address:

```
<cfcomponent>
  <cfproperty name="Number" type="numeric">
  <cfproperty name="Street" type="string">
  <cfproperty name="City" type="string">
  <cfproperty name="State" type="string">
  <cfproperty name="Country" type="string">
</cfcomponent>
```

This component represents a complex data type that can be used in a component that is exported as a web service, such as the following:
<cfcomponent>
  <cffunction name="echoAddress" returnType="address" access="remote">
    <cfargument name="input" type="address">
    <cfreturn arguments.input>
  </cffunction>
</cfcomponent>
**cfquery**

**Description**

Passes queries or SQL statements to a data source. Adobe recommends that you use the `cfqueryparam` tag within every `cfquery` tag, to help secure your databases from unauthorized users. For more information, see Security Bulletin ASB99-04, "Multiple SQL Statements in Dynamic Queries," in the Security Zone, [www.adobe.com/go/sn_asb99-04](http://www.adobe.com/go/sn_asb99-04), and Accessing and Retrieving Data in the Developing ColdFusion Applications.

**Category**

Database manipulation tags

**Syntax**

```xml
<cfquery
  name = "query name"
  blockFactor = "block size"
  cachedAfter = "date"
  cacheID = "ID"
  cacheRegion = "region"
  cachedWithin = "timespan"
  dataSource = "data source name"
  dbtype = "query"
  debug = "yes|no"
  fetchClientInfo = "yes|no"
  maxRows = "number"
  ormoptions = #orm options structure#
  password = "password"
  result = "result name"
  timeout = "seconds"
  username = "user name">
</cfquery>
```

⚠️ **Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cfdbinfo, cfinsert, cfprocparam, cfprocresult, cfqueryparam, cfstoredproc, cfttransaction, cfupdate; Optimizing database use in Optimizing ColdFusion applications in the Developing ColdFusion Applications

**History**

ColdFusion 10: Added the following attributes: `fetchClientInfo, cacheID, cacheRegion, clientInfo`}

ColdFusion 9.0.1: Introduced support for HQL queries; added the attribute `ormoptions`.

ColdFusion 9: Datasource attribute is optional now.
ColdFusion 8: Added the result variable that specifies the ID of a row.

ColdFusion MX 7:
- Added the result attribute for specifying an alternate name for the structure that holds the result variables.
- Added result variables for the SQL statement executed (sql), the number of records returned (recordcount), whether the query was cached (cached), an array of cfqueryparam values (sqlparameters), and the list of columns in the returned query (columnlist).
- Changed Query of Queries behavior: it now supports a larger subset of standard SQL.
- Changed dot notation support: ColdFusion now supports dot notation within a record set name. ColdFusion interprets such a name as a structure.
- Deprecated the connectString, dbName, dbServer, provider, providerDSN, and sql attributes, and all values of the dbtype attribute except query. They do not work, and might cause an error, in releases later than ColdFusion 5.
- No longer supports native drivers. It now uses JDBC (and ODBC-JDBC bridge) for database connectivity.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Name of query. Used in page to reference query record set. Must begin with a letter. Can include letters, numbers, and underscores.</td>
</tr>
<tr>
<td>blockFactor</td>
<td>Optional</td>
<td>1</td>
<td>Maximum rows to get at a time from server. Range: 1 - 100. Might not be supported by some database systems.</td>
</tr>
<tr>
<td>cachedAfter</td>
<td>Optional</td>
<td></td>
<td>Date value (for example, April 16, 1999, 4-16-99). If date of original query is after this date, ColdFusion uses cached query data. To use cached data, current query must use same SQL statement, data source, query name, user name, password. A date/time object is in the range 100 AD-9999 AD. When specifying a date value as a string, enclose it in quotation marks.</td>
</tr>
<tr>
<td>cacheID</td>
<td>Optional</td>
<td>ID to be used to store query result in cache. This ID can be used to either retrieve or remove query from cache.</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>cacheRegion</td>
<td>Optional</td>
<td>Cache region to be used to cache query result. If not specified, by default query is cached in the QUERY region.</td>
<td></td>
</tr>
<tr>
<td>cachedWithin</td>
<td>Optional</td>
<td>Timespan, using the CreateTimeSpan function. If original query date falls within the time span, cached query data is used. CreateTimeSpan defines a period from the present, back. Takes effect only if query caching is enabled in the Administrator. To use cached data, the current query must use the same SQL statement, data source, query name, user name, and password.</td>
<td></td>
</tr>
<tr>
<td>clientInfo</td>
<td>Optional</td>
<td>Structure containing properties of the client to be set on the database connection.</td>
<td></td>
</tr>
<tr>
<td>dataSource</td>
<td>Optional</td>
<td>TheDatasource attribute is now optional. If omitted, the query uses the datasource specified in the application. If it is not specified in either places, then the error will be thrown.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td><code>dbtype</code></td>
<td>Optional</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Results of a query as input. Specify either <code>dbtype</code> or <code>dataSource</code>. ColdFusion supports HQL in <code>cfquery</code>. Therefore, you can specify <code>dbtype=&quot;hql&quot;</code> as shown in the following example: <code>&lt;cfquery dbtype=&quot;hql&quot; name=&quot;artists&quot; ormoptions=#{cacheName=&quot;&quot;}#&gt;from Artists where firstname=&lt;cfqueryparam value=&quot;Aiden&quot;&gt;&lt;/cfquery&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>debug</code></td>
<td>Optional; value and equals sign may be omitted</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|           | • yes, or if omitted: if debugging is enabled, but the Administrator Database Activity option is not enabled, displays SQL submitted to the data source and number of records returned by query.  
|           | • no: if the Administrator Database Activity option is enabled, suppresses display. |
| `fetchClientInfo` | Optional | no |
|           | If set `yes`, returns a struct with the key-value pair passed by the last query. |
| `maxRows` | Optional | -1 (All) |
|           | Maximum number of rows to return in record set. |
| `ormoptions` | Optional |       |
|           | A struct that takes orm options for executing HQL. Applies only if `dbtype` is set to `hql`. |
| `password` | Optional |       |
|           | Overrides the password in the data source setup. |
result | Optional | Name for the structure in which cfquery returns the result variables. For more information, see Usage.

timeout | Maximum number of seconds that each action of a query is permitted to execute before returning an error. The cumulative time may exceed this value. For JDBC statements, ColdFusion sets this attribute. For other drivers, see the driver documentation.

username | Optional | Overrides user name in the data source setup.

Usage

Use this tag to execute a SQL statement against a ColdFusion data source. Although you can use the cfquery tag to execute any SQL Data Definition Language (DDL) or Data Manipulation Language (DML) statement, you typically use it to execute a SQL SELECT statement.

⚠️ Note

To call a stored procedure, use the cfstoredproc tag.

This tag creates a query object, providing this information in query variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query_name.currentRow</td>
<td>Current row of query that cfoutput is processing.</td>
</tr>
<tr>
<td>query_name.columnList</td>
<td>Comma-separated list of the query columns.</td>
</tr>
<tr>
<td>query_name.RecordCount</td>
<td>Number of records (rows) returned from the query.</td>
</tr>
</tbody>
</table>

The cfquery tag also returns the following result variables in a structure. You can access these variables with a prefix of the name you specified in the result attribute. For example, if you assign the name myResult to the result attribute, you would retrieve the name of the SQL statement that was executed by accessing myResult.sql #. The result attribute provides a way for functions or CFCs that are called from multiple pages, possibly at the same time, to avoid overwriting results of one call with another. The result variable of INSERT queries contains a key-value pair that is the automatically generated ID of the inserted row; this is available only for databases that support this feature. If more than one record was inserted, the value can be a list of IDs. The key name is database-specific.
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result_name.sql</td>
<td>The SQL statement that was executed.</td>
</tr>
<tr>
<td>result_name.recordcount</td>
<td>Number of records (rows) returned from the query.</td>
</tr>
<tr>
<td>result_name.cached</td>
<td>True if the query was cached; False otherwise.</td>
</tr>
<tr>
<td>result_name.sqlparameters</td>
<td>An ordered Array of cfqueryparam values.</td>
</tr>
<tr>
<td>result_name.columnList</td>
<td>Comma-separated list of the query columns.</td>
</tr>
<tr>
<td>result_name.ExecutionTime</td>
<td>Cumulative time required to process the query.</td>
</tr>
<tr>
<td>result_name.IDENTITYCOL</td>
<td>SQL Server only. The ID of an inserted row.</td>
</tr>
<tr>
<td>result_name.ROWID</td>
<td>Oracle only. The ID of an inserted row. This is not the primary key of the row, although you can retrieve rows based on this ID.</td>
</tr>
<tr>
<td>result_name.SYB.IDENTITY</td>
<td>Sybase only. The ID of an inserted row.</td>
</tr>
<tr>
<td>result_name.SERIAL_COL</td>
<td>Informix only. The ID of an inserted row.</td>
</tr>
<tr>
<td>result_name.GENERATED_KEY</td>
<td>MySQL only. The ID of an inserted row. MySQL 3 does not support this feature.</td>
</tr>
<tr>
<td>result_name.GENERATEDKEY</td>
<td>Supports all databases. The ID of an inserted row.</td>
</tr>
</tbody>
</table>

You can cache query results and execute stored procedures. For information about this and about displaying cfquery output, see the Developing ColdFusion Applications. Because the timeout attribute only affects the maximum time for each suboperation of a query, the cumulative time may exceed its value. To set a timeout for a page that might get a very large result set, set the Administrator > Server Settings > Timeout Requests option to an appropriate value or use the RequestTimeout attribute of the cfsetting tag (for example, `<cfsetting request Timeout="300">`). The Caching page of the ColdFusion Administrator specifies the maximum number of cached queries. Setting this value to 0 disables query caching.

You cannot use ColdFusion reserved words as query names. You cannot use SQL reserved words as variable or column names in a Query of Queries, unless they are escaped. The escape character is the bracket []; for example:

```
SELECT [count] FROM MYTABLE.
```

For a list of reserved keywords in ColdFusion, see Escaping reserved keywords in Query of Queries user guide in the Developing ColdFusion Applications.

Example

```
<!-- This example shows the use of CreateTimeSpan with CFQUERY ---->
<!-- to cache a record set. Define startrow and maxrows to ---->
<!-- facilitate 'next N' style browsing. ------>
<cfparam name="MaxRows" default="10">
<cfparam name="StartRow" default="1">
<!-- Query database for information if cached database information has
not been updated in the last six hours; otherwise, use cached data. -->
<cfquery
name="GetParks" datasource="cfdocexamples"
cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
SELECT PARKNAME, REGION, STATE
FROM Parks
ORDER BY ParkName, State
</cfquery>
<!-- Build HTML table to display query. ------------------------->
<table cellpadding="1" cellspacing="1">
<tr>
<td bgcolor="f0f0f0">
   
</td>
<td bgcolor="f0f0f0">
<b><i>Park Name</i></b>
</td>
<td bgcolor="f0f0f0">
<b><i>Region</i></b>
</td>
<td bgcolor="f0f0f0">
<b><i>State</i></b>
</td>
</tr>
<!-- Output the query and define the startrow and maxrows parameters.
Use the query variable CurrentCount to keep track of the row you are displaying. ---->
<cfoutput query="GetParks" startrow="#StartRow#" maxrows="#MaxRows#">
<tr>
<td valign="top" bgcolor="ffffed">
   #GetParks.CurrentRow#
</td>
<td valign="top">
   #ParkName#
</td>
<td valign="top">
   #Region#
</td>
<td valign="top">
   #State#
</td>
</tr>
</cfoutput>
<!-- If the total number of records is less than or equal to the total number of
rows, then offer a link to the same page, with the startrow value incremented by maxrows
(in the case of this example, incremented by 10). -------->
<tr>
<td colspan="4">
<cfif (StartRow + MaxRows) LTE GetParks.RecordCount>
<cfoutput><a href="#CGI.SCRIPT_NAME#?startrow=#Evaluate(StartRow + MaxRows)#">
See next `#MaxRows#` rows</a></cfoutput>
</cfif>
</td>
</tr>
</table>
cfqueryparam

Description

Verifies the data type of a query parameter and, for DBMSs that support bind variables, enables ColdFusion to use bind variables in the SQL statement. Bind variable usage enhances performance when executing a cfquery statement multiple times.

This tag is nested within a cfquery tag, embedded in a query SQL statement. If you specify optional parameters, this tag performs data validation. Adobe recommends that you use the cfqueryparam tag within every cfquery tag to help secure your databases from unauthorized users. For more information, see Security Bulletin ASB99-04, "Multiple SQL Statements in Dynamic Queries," at [www.adobe.com/go/sn_asb99-04](http://www.adobe.com/go/sn_asb99-04), and Accessing and Retrieving Data in the Developing ColdFusion Applications.

Category

Database manipulation tags

Syntax

```cfquery
<cfquery
  name = "query name"
  dataSource = "data source name"
  ...other attributes...
  SQL STATEMENT column_name =
  <cfqueryparam value = "parameter value"
    CFSQLType = "parameter type"
    list = "yes|no"
    maxLength = "maximum parameter length"
    null = "yes|no"
    scale = "number of decimal places"
    separator = "separator character">
  AND/OR ...additional criteria of the WHERE clause...
</cfquery>
```

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfinsert, cfproccparam, cfprocresult, cfquery, cfstoredproc, cftransaction, cfupdate; Enhancing security with cfqueryparam in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th><strong>value</strong></th>
<th><strong>Required</strong></th>
<th>Value that ColdFusion passes to the right of the comparison operator in a where clause. If CFSQLType is a date or time option, ensure that the date value uses your DBMS-specific date format. Use the CreateODate or DateFrom and Time Format functions to format the date value.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CFSQLType</strong></td>
<td><strong>Optional</strong></td>
<td><strong>CF_SQL_CHAR</strong></td>
</tr>
<tr>
<td><strong>list</strong></td>
<td><strong>Optional</strong></td>
<td>no</td>
</tr>
</tbody>
</table>
|  |  | • yes: the value attribute value is a delimited list.  
<p>|  |  | • no |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxLength</td>
<td>Optional</td>
<td>Length of string in <code>value</code> attribute</td>
</tr>
<tr>
<td>null</td>
<td>Optional</td>
<td>Whether parameter is passed as a null value: <code>yes</code>: tag ignores the <code>value</code> attribute. <code>no</code></td>
</tr>
<tr>
<td>scale</td>
<td>Optional</td>
<td>Number of decimal places in parameter. Applies to <code>CF_SQL_NUMERIC</code> and <code>CF_SQL_DECIMAL</code>.</td>
</tr>
<tr>
<td>separator</td>
<td>Required, if you specify a list in <code>value</code> attribute</td>
<td>Character that separates values in list, in <code>value</code> attribute.</td>
</tr>
</tbody>
</table>

### Usage

Use the `cfqueryparam` tag in any SQL statement (for example, SELECT, INSERT, UPDATE, and DELETE) that uses ColdFusion variables. For maximum validation of string data, specify the `maxlength` attribute.

This tag does the following:

- Allows the use of SQL bind parameters, which improves performance.
- Ensures that variable data matches the specified SQL type.
- Allows long text fields to be updated from a SQL statement.
- Escapes string variables in single-quotation marks.

To benefit from the enhanced performance of bind variables, use `cfqueryparam` for all ColdFusion variables, and your DBMS must support bind variables. ColdFusion validates and substitutes the validated parameter value back into the string. If validation fails, it returns an error message.

The validation rules are as follows:

- For these types, a data value can be converted to a numeric value: `CF_SQL_SMALLINT`, `CF_SQL_INTEGER`, `CF_SQL_REAL`, `CF_SQL_FLOAT`, `CF_SQL_DOUBLE`, `CF_SQL_TINYINT`, `CF_SQL_MONEY`, `CF_SQL_MONEY4`, `CF_SQL_DECIMAL`, `CF_SQL_NUMERIC`, and `CF_SQL_BIGINT`.
- For these types, a data value can be converted to a date supported by the target data source: `CF_SQL_DATE`, `CF_SQL_TIME`, `CF_SQL_TIMESTAMP`.
- For all other types, if the `maxlength` attribute is used, a data value cannot exceed the maximum length specified.

ColdFusion debug output shows the bind variables as question marks and lists the values beneath the query, in order of usage.
**Note**

To insert an empty string into a Microsoft Access table using the SequelLink ODBC Socket or SequelLink Access driver, the CFSQTYType attribute must specify CF_SQL_LONGVARCHAR.

The following table shows the mapping of ColdFusion SQL data types with JDBC SQL types and those of the listed database management systems:

<table>
<thead>
<tr>
<th>ColdFusion</th>
<th>JDBC</th>
<th>DB2</th>
<th>Informix</th>
<th>Oracle</th>
<th>MSSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF_SQL_ARRAY</td>
<td>ARRAY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF_SQL_BIGINT</td>
<td>BIGINT</td>
<td>Bigint</td>
<td>int8, serial8</td>
<td></td>
<td>bigint</td>
</tr>
<tr>
<td>CF_SQL_BINARY</td>
<td>BINARY</td>
<td>Char for Bit Data</td>
<td></td>
<td>binarytimestamp</td>
<td></td>
</tr>
<tr>
<td>CF_SQL_BIT</td>
<td>BIT</td>
<td></td>
<td>boolean</td>
<td></td>
<td>bit</td>
</tr>
<tr>
<td>CF_SQL_BLOB</td>
<td>BLOB</td>
<td>Blob</td>
<td>blob</td>
<td></td>
<td>blob, bfile</td>
</tr>
<tr>
<td>CF_SQL_CHAR</td>
<td>CHAR</td>
<td>Char</td>
<td>char, nchar</td>
<td></td>
<td>char</td>
</tr>
<tr>
<td>CF_SQL_CLOB</td>
<td>CLOB</td>
<td>Clob</td>
<td>clob</td>
<td></td>
<td>clob, nclob</td>
</tr>
<tr>
<td>CF_SQL_DATE</td>
<td>DATE</td>
<td>Date</td>
<td>date, datetime, year to day</td>
<td></td>
<td>date</td>
</tr>
<tr>
<td>CF_SQL_DECIMAL</td>
<td>DECIMAL</td>
<td>Decimal</td>
<td>decimal, money</td>
<td>number</td>
<td>decimal</td>
</tr>
<tr>
<td>CF_SQL_DISTINCT</td>
<td>DISTINCT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF_SQL_DOUBLE</td>
<td>DOUBLE</td>
<td>Double</td>
<td></td>
<td></td>
<td>double</td>
</tr>
<tr>
<td>CF_SQL_FLOAT</td>
<td>FLOAT</td>
<td>Float</td>
<td>float</td>
<td>number</td>
<td>real</td>
</tr>
<tr>
<td>CF_SQL_IDSTAMP</td>
<td>CHAR</td>
<td>Char</td>
<td>char, nchar</td>
<td></td>
<td>char</td>
</tr>
<tr>
<td>CF_SQL_INTEGER</td>
<td>INTEGER</td>
<td>Integer</td>
<td>integer, serial</td>
<td></td>
<td>integer</td>
</tr>
<tr>
<td>CF_SQL_LONG</td>
<td>LONGVARBINARY</td>
<td>Long Varchar for Bit Data</td>
<td>byte</td>
<td>long raw</td>
<td>longvarbinary</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>----------------------------</td>
<td>------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td>CF_SQL_LONG</td>
<td>LONGVARCHAR</td>
<td>Long Varchar</td>
<td>text</td>
<td>long</td>
<td>longvarchar</td>
</tr>
<tr>
<td>CF_SQL_MONEY</td>
<td>DOUBLE</td>
<td>Double</td>
<td></td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>CF_SQL_MONEY4</td>
<td>DOUBLE</td>
<td>Double</td>
<td></td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>CF_SQL_NULL</td>
<td>NULL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF_SQL_NUMERIC</td>
<td>NUMERIC</td>
<td>Numeric</td>
<td></td>
<td>numeric</td>
<td></td>
</tr>
<tr>
<td>CF_SQL_OTHE R</td>
<td>OTHER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF_SQL_REAL</td>
<td>REAL</td>
<td>Real</td>
<td>smallfloat</td>
<td>real</td>
<td></td>
</tr>
<tr>
<td>CF_SQL_REFCURSOR</td>
<td>REF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF_SQL_SMALLINT</td>
<td>SMALLINT</td>
<td>Smallint</td>
<td>smallint</td>
<td>smallint</td>
<td></td>
</tr>
<tr>
<td>CF_SQL_STRUCT</td>
<td>STRUCT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF_SQL_TIME</td>
<td>TIME</td>
<td>Time</td>
<td>datetime hour to second</td>
<td>time</td>
<td></td>
</tr>
<tr>
<td>CF_SQL_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Timestamp</td>
<td>datetime year to fraction(5), datetime year to second</td>
<td>date</td>
<td>timestamp</td>
</tr>
<tr>
<td>CF_SQL_TINYINT</td>
<td>TINYINT</td>
<td></td>
<td></td>
<td>tinyint</td>
<td></td>
</tr>
<tr>
<td>CF_SQL_VARBINARY</td>
<td>VARBINARY</td>
<td>Rowid</td>
<td>raw</td>
<td>varbinary</td>
<td></td>
</tr>
<tr>
<td>CF_SQL_VARCHAR</td>
<td>VARCHAR</td>
<td>Varchar</td>
<td>varchar, varchar, lvarchar</td>
<td>varchar2, varchar2</td>
<td>varchar</td>
</tr>
</tbody>
</table>
Example

<!---- This example shows cfqueryparam with VALID input in Course_ID. ---->
<h3>cfqueryparam Example</h3>
<cfset Course_ID = 12>
<cfquery name = "getFirst" dataSource = "cfdocexamples">
  SELECT *
  FROM courses
  WHERE Course_ID = <cfqueryPARAM value = "#Course_ID#" CFSQLType = 'CF_SQL_INTEGER'>
</cfquery>
<cfoutput query = "getFirst">
  <p>Course Number: #Course_ID#<br> Description: #descript#</p>
</cfoutput>

<!---- This example shows the use of CFQUERYPARAM when INVALID string data is in Course_ID. ---->
<p>This example throws an error because the value passed in the CFQUERYPARAM tag exceeds the MAXLENGTH attribute</p>
<cfset LastName="Peterson; DELETE employees WHERE LastName='Peterson'">
<!------- Note that for string input you must specify the MAXLENGTH attribute for validation. ------------------------------------------>
<cfquery name="getFirst" datasource="cfdocexamples">
  SELECT *
  FROM employees
  WHERE LastName=<cfqueryparam value="#LastName#" cfsqltype="CF_SQL_VARCHAR" maxlength="17">
</cfquery>
<cfoutput query="getFirst">
  <p>Course Number: #FirstName# #LastName#
  Description: #Department#</p></cfoutput>
Tags r-s

This section lists tags starting with alphabets R-S with their description.

- cfregistry
- cfreport
- cfreportparam
- cfrethrow
- creturn
- cfsavecontent
- cfschedule
- cfscript
- cfsearch
- cfselect
- cfset
- cfsetparam
- cfset
- cfstorage
- cfsharepoint
- cfsetting
- cfslider
- cfspreadsheet
- cfspydataset
- cfstoredproc
- cfswitch
**cfregistry**

**Description**

This tag is deprecated for the UNIX platform. Reads, writes, and deletes keys and values in the system registry. Provides persistent storage of client variables.

⚠️ **Note**

For this tag to execute, it must be enabled in the ColdFusion Administrator. For more information, see Configuring and Administering ColdFusion.

**Category**

*Other tags, Variable manipulation tags*

**Syntax**

The tag syntax depends on the `action` attribute value. See the following:

- `cfregistry action = "get"`
- `cfregistry action = "set"`
- `cfregistry action = "getAll"`
- `cfregistry action = "delete"`

**See also**

cfcookie, cfparam, cfsavecontent, cfschedule, cfset; About resource and sandbox security and Using Persistent Data and Locking in the Developing ColdFusion Applications

**History**

ColdFusion MX:

- Deprecated this tag on the UNIX platform. It might not work, and might cause an error, in later releases.
- Changed how persistent data is stored: ColdFusion now stores most persistent data outside the system registry, in XML files.

`cfregistry action = "getAll"`

**Description**

Returns all registry keys and values defined in a branch. You can access the values as you would any record set.

**Syntax**

```
<cfregistry
 action = "getAll"
 branch = "branch"
 name = "query name"
 sort = "asc|desc"
 type = "string|dWord|key|any">
```
**Note**

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

**See also**

[Using Persistent Data and Locking](#) in the *Developing ColdFusion Applications*

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Always <code>getall</code>.</td>
</tr>
<tr>
<td>branch</td>
<td>Required</td>
<td></td>
<td>Name of a registry branch.</td>
</tr>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Name of record set to contain returned keys and values.</td>
</tr>
<tr>
<td>sort</td>
<td>Optional</td>
<td><code>asc</code></td>
<td>Sorts query column data (case-insensitive). Sorts on Entry, Type, and Value columns as text. Specify a combination of columns from query output, in a comma-delimited list. For example: <code>sort = &quot;value desc, entry asc&quot;</code></td>
</tr>
</tbody>
</table>

- **asc**: ascending (a to z) sort order.
- **desc**: descending (z to a) sort order.
<table>
<thead>
<tr>
<th>type</th>
<th>Optional</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- string: returns string values.
- dWord: returns DWord values.
- key: returns keys.
- any: returns keys and values.

Usage

This tag returns `#entry#`, `#type#`, and `#value#` in a record set that you can access through tags such as `cfoutput`. To fully qualify these variables, use the record set name, as specified in the `name` attribute. If `#type#` is a key, `#value#` is an empty string. If you specify `type= "any"`, `getAll` also returns binary registry values. For binary values, the `#type#` variable contains UNSUPPORTED and `#value#` is blank.

Example

<!--- This example uses cfregistry with the getAll action. --->
<cfregistry action = "getAll"
    branch = "HKEY_LOCAL_MACHINE\Software\Microsoft\Java VM"
    type = "Any" name = "RegQuery">
    <h1>cfregistry action = "getAll"</h1>
    <cftable query = "RegQuery" colHeaders HTMLTable border = "yes">
        <cfcol header = "<b>Entry</b>" width = "35" text = "#RegQuery.Entry#"/>
        <cfcol header = "<b>Type</b>" width = "10" text = "#RegQuery.type#"/>
        <cfcol header = "<b>Value</b>" width = "35" text = "#RegQuery.Value#"/>
    </cftable>
</cfregistry>

cfregistry action = "get"

Description

Accesses a registry value and stores it in a ColdFusion variable.

Syntax

```
<cfregistry
    action = "get"
    branch = "branch"
    entry = "key or value"
    variable = "variable"
    type = "string|dWord|key">
```

⚠️ Note

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.
See also

Using Persistent Data and Locking in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Always get.</td>
</tr>
<tr>
<td>branch</td>
<td>Required</td>
<td></td>
<td>Name of a registry branch.</td>
</tr>
<tr>
<td>entry</td>
<td>Required</td>
<td></td>
<td>Registry value to access.</td>
</tr>
<tr>
<td>variable</td>
<td>Required</td>
<td></td>
<td>Variable into which to put value.</td>
</tr>
</tbody>
</table>
| type      | Optional| string  | • string: returns string value.  
                • dWord: returns DWord value.  
                • key: returns key's default value. |

Usage

If the value does not exist, the cfregistry tag does not create an entry.

Example

```cfm
<!--- This example uses cfregistry with the get action. --->
<cfregistry action = "get"
    branch = "HKEY_LOCAL_MACHINE\Software\Microsoft\Java VM"
    entry = "ClassPath" type = "String" variable = "RegValue">
<h1>cfregistry action = "get"
    <cfoutput>
    Java ClassPath value is #RegValue#
    </cfoutput>
<cfregistry action = "set"
    description="Add a registry key, adds a value, or updates a value."
    syntax="Description"
<cfregistry
  action = "set"
  branch = "branch"
  entry = "key or value"
  type = "string|dWord|key"
  value = "data">
</cfregistry>

**Note**

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also

Using Persistent Data and Locking in the Developing ColdFusion Applications

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td>Always</td>
<td>set.</td>
</tr>
<tr>
<td>branch</td>
<td>Required</td>
<td>Name of a registry branch.</td>
<td></td>
</tr>
<tr>
<td>entry</td>
<td>Required</td>
<td>Key or value to set.</td>
<td></td>
</tr>
</tbody>
</table>
| type      | Optional| ● string: sets a string value (default).
          |          | ● dWord: sets a DWord value.
          |          | ● key: creates a key. |
| value     | Optional| Value data to set. If you omit this attribute, the cfregistry tag creates default value, as follows: |
          |          | ● string: creates an empty string: ".". |
          |          | ● dWord: creates a value of 0 (zero). |

**Usage**

If it does not exist, the cfregistry tag creates the key or value.

**Example**
<cfregistry action = "delete">

Description

Deletes a registry key or value.

Syntax

```cfml
<cfregistry 
    action = "delete"
    branch = "branch"
    entry = "key or value">
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

Using Persistent Data and Locking in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>Always delete.</td>
</tr>
</tbody>
</table>
| branch    | Required|         | • For key deletion: name of registry key to delete. Do not specify the entry attribute.  
<p>|           |         |         | • For value deletion: name of registry branch that contains value to delete. Specify the entry attribute. |</p>
<table>
<thead>
<tr>
<th>entry</th>
<th>Required for value deletion</th>
<th>Value to delete</th>
</tr>
</thead>
</table>

**Usage**

If you delete a key, the `cfregistry` tag also deletes values and subkeys defined beneath it.

**Example**

```cfreg
<cfregistry action = "delete"
  branch = "HKEY_LOCAL_MACHINE\Software\cflangref\tempkey"
  entry = "LastCFM01">
<h1>cfregistry action = "delete"</h1>
```
cfreport

Description

Used to do either of the following:

- Execute a report created with the ColdFusion Report Builder, displaying it in PDF, Adobe FlashPaper, RTF, HTML, XML or Excel format. Optionally, you can save this report to a file.
- Run a predefined Crystal Reports report. Applies only to Windows systems.

Category

Data output tags

Syntax

ColdFusion Report Builder syntax:
<cfreport
  format = "PDF|FlashPaper|Excel|RTF|HTML|XML"
  template = "absolute pathname or pathname relative to the report file"
  encryption = "128-bit|40-bit|none"
  filename = "output filename"
  name = "ColdFusion variable"
  ownerpassword = "password"
  overwrite = "no|yes"
  permissions = "permission list"
  query = "query variable"
  resourceTimespan = #CreateTimeSpan (days, hours, minutes, seconds)#
  style = "CSS style definition or css file pathname"
  userpassword = "password">
  <cfreportparam ...
</cfreport>

Crystal Reports syntax:
<cfreport
  report = "report path"
  dataSource = "data source name"
  formula = "formula"
  orderBy = "result order"
  password = "password"
  timeout = "number of seconds"
  type = "standard|netscape|microsoft"
  username = "username">
  <cfreportparam ...
</cfreport>

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfcollection, cfdocument, cfdocumentitem, cfdocumentsection, cfexecute, cfindex, cfobject, cf
Creating Reports with Report Builder in the Developing ColdFusion Applications; Report Builder online Help

History

ColdFusion 8: Added the style and resourceTimespan attributes. Added the HTML and XML values to the format attribute.
ColdFusion MX 7.0.1: Added the RTF value to the format attribute, to let you generate reports in RTF format.
ColdFusion MX 7: Added support for the ColdFusion Report Builder.
ColdFusion MX: Changed data source connection behavior: Crystal Reports now establishes an independent connection to the data source. The connection is not subject to any ColdFusion data source-specific restrictions. For example, the Crystal Reports server can access a data source, regardless of whether it is disabled in the ColdFusion Administrator.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Applies to</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>Crystal Reports</td>
<td>Optional</td>
<td></td>
<td>Name of registered or native data source.</td>
</tr>
<tr>
<td>encryption</td>
<td>Report Builder</td>
<td>Optional</td>
<td>none</td>
<td>(format=&quot;PDF&quot; only) Type of encryption for the report output. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 128-bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 40-bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• none</td>
</tr>
<tr>
<td>filename</td>
<td>Report Builder</td>
<td>Optional</td>
<td>Filename to contain the report. You cannot specify both the name and filename attributes. The filename extension must match the value of the format attribute. If you write report output to an HTML file, ColdFusion automatically creates a directory relative to the output file in the format <code>filename_files</code>. Also, it generates PNG files for any charts in the report and copies of any image files imported into the report and stores them in this directory.</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>Report Builder</td>
<td>Required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Format of the report output:
- PDF
- FlashPaper
- Excel
- RTF
- XML
- HTML

When you write report output directly to the browser in HTML format, ColdFusion generates a temporary directory and files for the images in the report. The location of the temporary directory that contains the image files is: C:\ColdFusion9\tmpCache\CFFile\Servlet_cfreport_report_unique_identifier_To determine when the images are removed from the browser, use the `resourceTimespan` attribute.
<table>
<thead>
<tr>
<th><strong>formula</strong></th>
<th><strong>Crystal Reports</strong></th>
<th><strong>Optional</strong></th>
<th><strong>One or more named formulas. Terminate each formula with a semicolon. Use the format:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>formula = &quot;formulaname1 = 'formula1';formulaname2 = 'formula2';&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If you use a semicolon in a formula, escape it by typing it twice (;;). For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>formula = &quot;Name1 = 'Val_1a;;Val_1b';Name2 = 'Val2';&quot;</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td><strong>Report Builder</strong></td>
<td><strong>Optional</strong></td>
<td><strong>Name of the ColdFusion variable that contains the report output. You cannot specify both name and filename. This attribute is not valid when format=&quot;HTML&quot;.</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Name of the ColdFusion variable that contains the report output. You cannot specify both name and filename. This attribute is not valid when format=&quot;HTML&quot;.</td>
</tr>
<tr>
<td>orderBy</td>
<td>Crystal Reports</td>
<td>Optional</td>
<td>Orders results according to your specifications.</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>overwrite</td>
<td>Report Builder</td>
<td>Optional</td>
<td>yes/no Specifies whether to overwrite files that have the same name as that specified in the <code>filename</code> attribute:</td>
</tr>
<tr>
<td>ownerPassword</td>
<td>Report Builder</td>
<td>Optional</td>
<td>(format=&quot;PDF&quot; only) Owner password for the report,</td>
</tr>
<tr>
<td>password</td>
<td>Crystal Reports</td>
<td>Optional</td>
<td>Password that corresponds to username required for database access. Overrides default settings for data source in the ColdFusion Administrator.</td>
</tr>
<tr>
<td>permissions</td>
<td>Report Builder</td>
<td>Optional</td>
<td>(format=&quot;PDF&quot; only) Specifies one or more of the following permissions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- AllowPrinting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- AllowModifyContents</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- AllowCopy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- AllowModifyAnnotations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- AllowFillIn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- AllowScreenReaders</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- AllowAssembly</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- AllowDegradedPrinting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Separate multiple permissions with commas.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>query</td>
<td>Name of the query that contains input data for the report. This query overrides the query in the Report Builder report. The ColdFusion query must contain at least all of the columns included in the Report Builder query; however, the WHERE clause can differ. If you omit this parameter, Report Builder uses the data from the internal SQL statement or from c_freportparam items.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>report</td>
<td>Report path name. Store Crystal Reports files in the same directories as ColdFusion page files.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>ReportBuilder</td>
</tr>
<tr>
<td>report</td>
<td>Crystal Reports</td>
</tr>
<tr>
<td>resourceTimespan</td>
<td>Report Builder</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>

(resourceTimespan only) Life span of the resource directory. When you export a Report Builder report in HTML format, ColdFusion writes any images or other resource files in the report to a temporary resource directory. Use this attribute to determine when the resource directory is deleted. For the value, use the `CreateTimeSpan` function and specify the time span in days, hours, minutes, and seconds, separated by commas; for example, to delete the resource directory after one hour, specify: `#CreateTimeSpan(0,1,0,0)#`)

If the value is set to 0, the resource directory persists until the next server restart. ColdFusion deletes the resource directory only when `{{format="HTML"}}and neither the name nor filename attribute is specified. The default setting is 5 minutes: `#CreateTimeSpan(0,0,5,0)#`
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>style</td>
<td>ReportBuilder</td>
<td>Optional</td>
<td>Style in CSS format that overrides a style defined in the Report Builder report at run time. You can specify an absolute file path, a file path relative to the report, or a string in valid CSS format. For the styles to take effect, the style names must match Style Name attributes assigned to elements in the Report Builder report. You can generate a CSS file in Report Builder and export it or you can create a CSS file with a text editor. For a list of supported CSS styles, see the section Style properties.</td>
</tr>
<tr>
<td>template</td>
<td>Report Builder</td>
<td>Required</td>
<td>Specifies the pathname to the Report Builder (CFR) file, relative to the web root.</td>
</tr>
<tr>
<td>timeout</td>
<td>Crystal Reports</td>
<td>Optional</td>
<td>Specifies the maximum time, in seconds, in which a connection must be made to a Crystal Reports file.</td>
</tr>
</tbody>
</table>
| type     | Crystal Reports       | Optional| Type of report:  
  - standard (not valid for Crystal Reports 8.0)  
  - netscape  
  - microsoft |
userName | Crystal Reports | Optional | Username required for entry into a database from which the report is created. Overrides default settings for data source in ColdFusion Administrator.

userPassword | Report Builder | Optional | (format="PDF" only) User password.

Usage

Use this tag to generate a report using a report definition created in either ColdFusion Report Builder or in Crystal Reports. (For more information on using the ColdFusion Report Builder, display the online help by opening the Report Builder and pressing F1.)

⚠️ Note

The Excel report output format type provides limited support for the formatting options available in ColdFusion Reporting. Images and charts are not supported and numeric data containing formatting (commas, percents, currency, and so on) appear as plain text in Excel. The Excel output format supports simple reports only and Adobe recommends that you give careful design and layout consideration to reports designed for Excel output.

This tag requires an end tag.

Using Cascading Style Sheets

You can override Cascading Style Sheets (CSS) in Report Builder reports at run time by using the style attribute of the cfreport tag in ColdFusion.

You can create CSS files in one of two ways: by exporting styles with the Export Report Styles icon in Report Builder or by creating a CSS file in any text editor. For the CSS styles to take effect, however, you must use Report Builder to assign the style names to the elements in the report. (The exception is the default style: you can use the style attribute to define the default style in ColdFusion and apply it to the report even if the default style is not defined in Report Builder.) After you assign the style names in Report Builder, you can update the style definitions in the CSS file at any time and apply them at run time by using the cfreport and cfreportparam tags. If your report contains subreports, the default style applies to the master report and to all of the subreports. If the master report uses CSS styles other than the default style, the CSS styles do not apply to the subreports unless you specify them explicitly.

The following code shows how to apply three different style sheets to the main report and two subreports at run time:

```html
<cfreport template="myreport.cfr" style="mystyle.css" format="PDF">
<cfreportParam subreport="subreport1" style="subreport1-style.css">
<cfreportParam subreport="subreport2" style="subreport2-style.css">
</cfreport>
```
The following code shows how to apply a CSS style as a value of the `style` attribute:

```cfreport template="myreport.cfr" style='mystyle { defaultStyle: true; font-family:"Comic Sans MS"; color: ##00FF00; }' format="FlashPaper">
</cfreport>
```

The following code shows how to create a variable called `myStyle` and use it as a value of the `style` attribute:

```cfset mystyle='mystyle { defaultStyle: false; font-family: "Comic Sans MS"; }'>
<cfreport template="myreport.cfr" style="#mystyle#" format="HTML">
</cfreport>
```

### Style attribute syntax

The style file or string must be valid CSS syntax. For more information, see [http://www.w3.org/Style/CSS/](http://www.w3.org/Style/CSS/). The style must contain one or more rule sets. Each rule set consists of a simple selector, which is the Report Builder style name, followed by a declaration block, which consists of a series of declarations separated by semicolons. A declaration is a property:value pair. If a selector contains invalid syntax, ColdFusion ignores the selector and its declaration block. Selectors and properties not supported by this feature are ignored. Styles are case-insensitive, except parts not under the control of CSS (such as font names).

The following example shows the CSS definition for the default style:

```css
DefaultStyle
{
  default-style: true;
  color: black;
  font-family: Arial, "Comic Sans MS";
  font-size: 16;
  text-decoration: underline;
}
```

The following example shows the CSS definition for a custom style called `PurpleBoldItalicText`:

```css
PurpleBoldItalicText
{
  color: purple;
  font: italic bold 20px 30px Arial;
}
```

Identifiers for styles must be CSS2-compliant. For example, CSS1 allows `_` in identifiers, but CSS2 does not. In CSS2, identifiers, including element names, classes, and IDs in selectors, can contain only the characters A-Z, a-z, and 0-9. Also, they can include ISO 10646 characters 161 and higher and the hyphen character `-`; however, identifiers cannot start with a hyphen or a digit. They can contain escaped characters and any ISO 10646 character as a numeric code. For example, you can write the identifier “B&W?” as “B\&W?” or “B\26 W\3F”.
Style properties

The following table shows the style properties exported by Report Builder:

<table>
<thead>
<tr>
<th>Property name</th>
<th>Report Builder only</th>
<th>Valid values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background-color</td>
<td>No</td>
<td>See the section Valid color values</td>
<td>Background color of the specified report element, if the element is not transparent. The default background color is white.</td>
</tr>
<tr>
<td>border</td>
<td>No</td>
<td>border-width</td>
<td>A shorthand property that specifies the border-width, border-style, and border-color properties for all of the borders in an element.</td>
</tr>
<tr>
<td>border-color</td>
<td>No</td>
<td>See the section Valid color values</td>
<td>Border color for text, images, and charts. You can customize each side of the border. The default color is white.</td>
</tr>
<tr>
<td>border-style</td>
<td>No</td>
<td>solid, dashed, none</td>
<td>A shorthand property that specifies the border-top-style, border-right-style, border-bottom-style, and border-left-style (the comma-separated values must be in this order). If one or more values are not specified, the value for the opposite side is used. If only one value is listed, that value applies to all sides. The none value overrides the border-width value. Any other values, including hidden, dotted, groove, ridge, inset, outset, and double, are displayed as solid.</td>
</tr>
<tr>
<td>Property</td>
<td>Unit</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>border-top-color</td>
<td></td>
<td>Color of the element's top, left, bottom, and right border. For more details see that section <em>Border and margin styles</em>. If no border-color property is specified, the value of the color property is used instead.</td>
<td></td>
</tr>
<tr>
<td>border-top-style</td>
<td></td>
<td>Line style of the element's top, left, bottom, and right border. For more details see the section <em>Border and margin styles</em>. Any value other than dashed displays as solid.</td>
<td></td>
</tr>
</tbody>
</table>
| border-top-width            |      | Thickness of the top, left, bottom, and right border of an element. Negative values are not valid. For more details see the section *Border and margin styles*:  
  * thin = 1/2 pt  
  * medium = 2px  
  * thick = 4px |
| border-width                |      | A shorthand property that specifies the border-top-width, border-right-width, and border-bottom-width properties with a single property and value notation (the comma-separated values must be in this order). If one or more values are not specified, the value for the opposite side is used. If only one value is listed, it applies to all sides:  
  * thin = 1/2 pt**  
  * medium = 2px  
  * thick = 4px |

**Note:** 
- **thin** = 1/2 pt
- **medium** = 2px
- **thick** = 4px

**See the section Valid color values**
| clip     | No | auto stretch ratio | Specifies how images are resized:  
|          |    |                   | - auto: If the dimensions of the source image differ from the element in the report, this attribute crops the image to fit within the element boundaries. The image is not resized. Only the part of the image that fits in the boundaries is displayed.  
|          |    |                   | - stretch: If the dimensions of the source image differ from the element in the report, this attribute resizes the image so that it fits within the element boundaries. If the dimensions differ, the image is distorted.  
|          |    |                   | - ratio: If the dimensions of the source image differ from the element in the report, this attribute resizes the image to fit within the element boundaries but maintains the aspect ratio of the source so that the image is not distorted.  
| color    | No | See the section Valid color values | Color of the text (in text elements) and the border (in graphic elements). The default color is black.  
<p>| corner-radius | Yes | integer | Radius for arcs used to draw the corner of rectangles. The default is 0 (square corners). Values less than 0 are interpreted as 0. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default-style</td>
<td>Yes</td>
<td>Default style for elements that do not or cannot have a style applied. A subreport inherits its parent's default-style if it does not have one of its own.</td>
</tr>
<tr>
<td>embed-pdf-font</td>
<td>Yes</td>
<td>Specifies whether fonts are embedded in the PDF document. Embedded fonts insure that the fonts display properly even if the font is not installed on the system where the report is viewed.</td>
</tr>
<tr>
<td>empty-cells</td>
<td>No</td>
<td>Shows or hides a null value for text expressions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• show: If the text expression returns a null value, the string &quot;null&quot; is displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• hide: If the text expression returns a null value, the null value is replaced with an empty string. This is the default.</td>
</tr>
<tr>
<td>font</td>
<td>No</td>
<td>Font characteristic specifications. Use this as a shorthand to specifying multiple property values; for example: font: italic 20px Arial;D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>efault values for this property match those used for the individual properties. Default values for the individual properties are applied to the values omitted from the font property.</td>
</tr>
<tr>
<td>Property</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>font-family</td>
<td>No</td>
<td>Comma-separated list of font names. Group of fonts to apply to the element. The first font found in the list is applied to the element. The default is: <code>font-family: Arial, Helvetica, sansserif;</code> If a font name contains spaces, enclose the name in quotation marks, for example: <code>font-family: Courier, &quot;Courier New&quot;, Arial;</code></td>
</tr>
<tr>
<td>font-size</td>
<td>No</td>
<td><code>[length]</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Font size measured in points or pixels. Negative values are not valid. The default value is 10 points. You can specify points or pixels. 1 pixel = 0.75 points. This property also is a component of the font property. Standard CSS supports other types of values not supported by Report Builder.</td>
</tr>
<tr>
<td>font-style</td>
<td>No</td>
<td>normal italic oblique</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Font style. The italic and oblique values are similar. The default value is normal. Also, this property is a component of the font property.</td>
</tr>
<tr>
<td>font-weight</td>
<td>No</td>
<td>normal bold bolder lighter 100 200 300 400 500 600 700 800 900</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Font weight. Report Builder does not support varying degrees of boldness or lightness; therefore, normal and lighter appear as normal; all other values appear as bold. The default is normal. Also, this property is a component of the font property.</td>
</tr>
<tr>
<td>Property</td>
<td>Default</td>
<td>Values</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>line-height</td>
<td>No</td>
<td>normal, number, length, percentage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>line-size</td>
<td>Yes</td>
<td>none, thin, 1px, 2px, 4px, dashed</td>
</tr>
<tr>
<td>Property</td>
<td>Required</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>margin</td>
<td>No</td>
<td>top-integer, right-integer, bottom-integer, left-integer</td>
</tr>
<tr>
<td>margin-topmargin-leftmargin-bottommargin-right</td>
<td>No</td>
<td>integer</td>
</tr>
<tr>
<td>parent-style</td>
<td>Yes</td>
<td>styleName</td>
</tr>
<tr>
<td>text-align</td>
<td>No</td>
<td>left, center, right, justify</td>
</tr>
<tr>
<td>Property</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>text-decoration</td>
<td>No</td>
<td>Text characteristics not defined with the <code>font-style</code> and <code>font-weight</code> properties. The color of the text-decoration is determined by the color property for the element. Unknown values are ignored.</td>
</tr>
<tr>
<td>text-rotation</td>
<td>Yes</td>
<td>Rotation of text elements. Use it to change the text direction by rotating it 90 degrees to the right or left.</td>
</tr>
<tr>
<td>transparency-mode</td>
<td>Yes</td>
<td>Transparency of elements. Graphic elements, such as rectangles and lines, are opaque by default, but images are transparent. Subreport elements, static text, and text fields are transparent by default.</td>
</tr>
<tr>
<td>vertical-align</td>
<td>No</td>
<td>Alignment of text and images on the vertical axis. The default value is top.</td>
</tr>
<tr>
<td>Property</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>xhtml-formatted-text</td>
<td>Yes</td>
<td>Specifies whether the text element contains XHTML-compatible instructions:</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td>• true: The text element contains XHTML-compatible instructions, for example, <code>&lt;b&gt;My Text Label&lt;/b&gt;</code>. In this example, the text within the tags displays in bold (My Text Label), and the tags (<code>&lt;b&gt; &lt;/b&gt;</code>) are not displayed.</td>
</tr>
<tr>
<td></td>
<td>false</td>
<td>• false: The text element does not contain XHTML-compatible instructions; therefore, all the text within the text element is displayed. This is the default.</td>
</tr>
</tbody>
</table>

Styles or values that are not supported by Report Builder are ignored in the report, in which case, if a default-style is defined, Report Builder applies the default style to the element.

**Valid color values**

You can specify a color as #RRGGBB. This represents a color that uses a triplet of hexadecimal values concatenated together. The values represent the red, green, and blue components for a given color. The range of each component value is 00-FF in hexadecimal. Also, you can use one of the 140 X11 color names (see [http://www.blooberry.com/indexdot/color/x11makerFrameNS.htm](http://www.blooberry.com/indexdot/color/x11makerFrameNS.htm)). The color name is case-insensitive. This set of names assigns names to specific RGB values. Also, a color name can also be specified as ##RGB, rgb(r,g,b), or rgb(r%,g%,b%). See CSS Color Units for syntax details (see [http://www.blooberry.com/indexdot/css/syntax/units/color.htm](http://www.blooberry.com/indexdot/css/syntax/units/color.htm)). UI Name is not supported.

The following example shows the different ways you can represent the color lime:

```
color: lime
color: #00FF00
color: #0F0
color: rgb(0,255,0)
color: rgb(0%,50%,0%)
```

If you specify a color in hexadecimal format as part of the `style` attribute for the `cfreport` tag, use the format `##00FF00`. For example:
<cfreport template="myreport.cfr" style='mystyle { defaultStyle: true; font-family:"Comic Sans MS"; color: #00FF00; }' format="HTML"/>

**Border and margin styles**

Use the `border-width`, `border-style`, `border-color`, and `margin` properties when all four sides of the element have the same value. You can specify from one to four parameters for these properties:

<table>
<thead>
<tr>
<th>Number of parameters</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>border-width: thick</td>
<td>Parameter applied to all four sides of the element's border.</td>
</tr>
<tr>
<td>2</td>
<td>border-width: thick, thin</td>
<td>First parameter (thick) applied to the top and bottom sides; second parameter (thin) applied to the left and right sides.</td>
</tr>
<tr>
<td>3</td>
<td>border-width: thick, thin, medium</td>
<td>First parameter (thick) applied to the top; second parameter (thin) applied to the left and right sides; third parameter (medium) applied to the bottom.</td>
</tr>
<tr>
<td>4</td>
<td>border-width: thick, thin, medium, thick</td>
<td>First parameter (thick) applied to the top; second parameter (thin) applied to the right side; third parameter (medium) applied to the bottom; fourth parameter (thick) applied to the left side.</td>
</tr>
</tbody>
</table>

You can use the properties for each side of a border to override the style specified by the `border-width`, `border-style`, `border-color`, and `margin` properties.

**Example**

**Example 1:** This example shows the use of cfreport for the ColdFusion Report Builder.

```cfquery name="northwindemployees" datasource="localnorthwind">
    SELECT EmployeeID, LastName, FirstName, Title, City, Region, Country
    FROM Employees
    ORDER BY Country, City
</cfquery>

<CFREPORT format="PDF" template="FifthReport.cfr" query="#northwindemployees#"/>
Example 2: This view-only example shows the use of cfreport for Crystal Reports.

```cfml
<h3>cfreport Tag</h3>
<p>cfreport lets reports from the Crystal Reports Professional report writer display through a ColdFusion interface. To run, the tag requires the name of the report. cfreport can also pass information to the report file displayed, to change the output conditions.</p>
<p>This example would run a report called "monthlysales.rpt" and pass it an optional filter condition to show only the information for a subset of the report.</p>

<cfreport report = '/reports/monthlysales.rpt'>
  {Departments.Department} = 'International'
</cfreport>

<p>Substitute your report files and filters for this code. cfreport can put Crystal Reports into web pages.</p>
```
The `cfreportparam` tag lets you perform the following tasks:

- Pass input parameters to a ColdFusion Report Builder report definition.
- Override query data in subreports and charts defined in Report Builder reports.
- Override styles defined in Report Builder subreports.

The `cfreportparam` tag is always a child tag of the `cfreport` tag.

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chart</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>query</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>series</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>style</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>subreport</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Note

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

### See also

- `cfreport`: [Creating Reports with Report Builder](https://help.adobe.com/Developing/ColdFusion Applications) in the *Developing ColdFusion Applications*; Report Builder online Help

### History

- **ColdFusion 8**: Added the `chart`, `query`, `series`, `subreport`, and `style` attributes.
- **ColdFusion MX 7**: Added this tag.
<p>| <strong>chart</strong> | <strong>Optional</strong> | Name of the chart contained in a report or subreport. The value of this attribute must match Name property of a chart defined in the Report Builder report. If you specify the <code>chart</code> attribute, you cannot specify the <code>subreport</code> or <code>name</code> attribute. |
| <strong>name</strong> | <strong>Optional</strong> | Variable name for data that is passed. The value of this attribute must match the name of an input parameter defined in the Report Builder report. If you specify the <code>name</code> attribute, you cannot specify the <code>chart</code> or <code>subreport</code> attribute. |
| <strong>query</strong> | <strong>Optional</strong> | Query value to pass to a subreport or chart. The ColdFusion query must contain at least all of the columns included in the Report Builder query. Charts and subreports require this attribute. |
| <strong>series</strong> | <strong>Optional</strong> | Ordinal number of a chart series to use for the query. This attribute is valid only when the <code>chart</code> attribute is specified. |
| <strong>subreport</strong> | <strong>Optional</strong> | Name of the subreport. The value of this attribute must match the Name property of the subreport in Report Builder. Subreport names within a report must be unique. If you specify the <code>subreport</code> attribute, you cannot specify the <code>chart</code> or <code>name</code> attribute. |</p>
<table>
<thead>
<tr>
<th>attribute</th>
<th>status</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>style</td>
<td>Optional</td>
<td>Style in CSS format for a subreport. The value can be an absolute file path, a file path relative to the report, or a string in valid CSS format. For the styles to take effect, the style names must match Style Name attributes assigned to elements in the Report Builder report. You can generate the CSS file in Report Builder and exported or created with a text editor. For a list of supported CSS styles, see the section Style properties.</td>
</tr>
<tr>
<td>value</td>
<td>Optional (see Description)</td>
<td>Value of the data that is sent. Specify the value attribute with the name attribute. You cannot specify this attribute when a chart or subreport attribute is specified. The value can be a string or a variable.</td>
</tr>
</tbody>
</table>

**Usage**

You can specify only one of the following attributes in a `cfreportparam` tag:

- name
- subreport
- chart

You can use the `query`, `subreport`, and `chart` attributes to override Report Builder queries and chart information at run time. This way you can customize subreport and chart data from the CFM page without having to change the queries built into your report.

For example, in Report Builder, you can create a master report that contains several subreports and populate each subreport with a different query. Instead of modifying the queries in Report Builder, you can customize your reports by creating modified queries on the ColdFusion calling page. The ColdFusion query must contain at least all of the columns included in the Report Builder query.

**Note**

You cannot specify a subreport query that depends on arguments from the master report. Instead, you can define a CFML function or CFC method that returns the subreport query given the arguments from the master report. ColdFusion calls this code when it executes the subreport.

On the calling CFM page, you can specify a `cfreportparam` tag for any subreport and chart in the Report.
Builder report. The value of the `subreport` or `chart` attribute must match the Name property of the subreport or chart defined in the Report Builder report. (Charts are treated like subreports.) The following code shows a master report that contains two subreports and a chart with two chart series:

```cfml
<cfreport template="myreport.cfr" query="master" format="RTF">
  <cfreportParam subreport="subreport1" query="subquery1">
  <cfreportParam subreport="subreport2" query="subquery2">
    <cfreportParam chart="chart1" series="1" query="chartquery1">
    <cfreportParam chart="chart1" series="2" query="chartquery2">
      <cfreportParam name="ReportDate" value="#DateFormat(Now())#, #TimeFormat(Now())#">
      </cfreport>
  </cfreport>
</cfreport>
```

The `cfreportparam` tag also lets you override CSS styles assigned to subreports in Report Builder. Use the `style` attribute with the `subreport` attribute; the value of the `subreport` attribute must match the name of the subreport in Report Builder. The following code applies a style sheet to the master report and two different style sheets to the subreports:

```cfml
<cfreport template="myreport.cfr" style="myStyle.css" format="PDF">
  <cfreportParam subreport="subreport1" style="subreport-style.css">
  <cfreportParam subreport="subreport2" style="subreport-style.css">
  </cfreport>
</cfreport>
```

For more information, see the section *Using Cascading Style Sheets.*

Example

```cfml
<!---- The following example shows how to override a query in a Report Builder report from the CFM page. The `cfreportparam` tag adds the current date and time to the report.---->
<cfquery name="coursedept" datasource="cfdocexamples">
  SELECT Departments.Dept_ID as dDept_ID, Departments.Dept_Name, CourseList.Course_ID, CourseList.Dept_ID as cDept_ID, CourseList.CorNumber, CourseList.CorName, CourseList.CorLevel 
  FROM Departments, CourseList 
  WHERE Departments.Dept_ID = CourseList.Dept_ID 
  ORDER BY CourseList.Dept_ID
</cfquery>

<cfreport format="PDF" template="FourthReport.cfr" query="#coursedept#" overwrite="yes">
  <cfreportparam name="ReportTime" value="#DateFormat(Now())#, #TimeFormat(Now())#">
  </cfreport>
</cfreport>
```
cfrethrow

Description

Rethrows the currently active exception. Preserves the exception's cfcatch.type and cfcatch.tagContext variable values.

Category

Exception handling tags, Extensibility tags

Syntax

<cfrethrow>

See also

cferror, cfthrow, cftry; Handling runtime exceptions with ColdFusion tags in the Developing ColdFusion Applications

Usage

Use this tag within a cfcatch block. This tag is useful in error handling code, if the error handler cannot handle an error that it catches. For example, if cfcatch type = "any" gets a DATABASE exception, and the code is designed to handle only CFX exceptions, the handler raises the exceptions again, with details intact, so that a higher-level handler can process the error information. If you used the cfthrow tag, the type and details of the original exception would be lost.

Example
<h3>cfrethrow Example</h3>

<!--- Rethrow a DATABASE exception. --->
<cftry>
  <cfquery name = "GetMessages" dataSource = "cfdocexamples">
    SELECT*
    FROM Messages
  </cfquery>
  <cfcatch type = "DATABASE">
    <!--- If database signalled a 50555 error, ignore; otherwise, rethrow exception. --->
    <cfif cfcatch.sqlstate neq 50555>
      <cfrethrow>
    </cfif>
  </cfcatch>
</cftry>

<h3>Sorry, this request can't be completed</h3>

<h4>Catch variables</h4>
<cfoutput>
  <cfloop collection = #cfcatch# item = "c">
    <br>
    <cfif IsSimpleValue(cfcatch[c])>
      #c# = #cfcatch[c]#
    </cfif>
  </cfloop>
</cfoutput>
**cfreturn**

**Description**

Returns result values from a component method. Contains an expression returned as result of the function.

**Return value**

An expression; the result of the function from which this tag is called.

**Category**

Extensibility tags

**Syntax**

```xml
<cfreturn
  expr>
</cfreturn>
```

**See also**

cfargument, cfcomponent, cffunction, cfinvoke, cfinvokeargument, cfobject, cfproperty; Building and Using ColdFusion Components in the Developing ColdFusion Applications

**History**

ColdFusion MX: Added this tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expr</td>
<td>Required</td>
<td></td>
<td>Function result; value of any type.</td>
</tr>
</tbody>
</table>

**Usage**

This tag is equivalent to a return statement within a cfscript tag. It accepts one return variable argument. To return more than one value, populate a structure with name-value-pairs, and return the structure with this tag. To access the result value from this tag, you use the variable scope that is the value of the cfinvoke tag returnV ariable attribute.

You can code a maximum of one cfreturn tag within a function.

For example code, see Building and Using ColdFusion Components in the Developing ColdFusion Applications.

**Example**

...
<cfcomponent>
<cffunction name="getEmp">
   <cfquery name="empQuery" datasource="ExampleApps">
      SELECT FIRSTNAME, LASTNAME, EMAIL
      FROM tblEmployees
   </cfquery>
   <cfreturn empQuery>
</cffunction>
<cffunction name="getDept">
   <cfquery name="deptQuery" datasource="ExampleApps">
      SELECT *
      FROM tblDepartments
   </cfquery>
   <cfreturn deptQuery>
</cffunction>
</cfcomponent>
CFSaveContent

Description

Saves the generated content of the cfsavecontent tag, including the results of evaluating expressions and executing custom tags, in the specified variable.

Category

Variable manipulation tags

Syntax

```
<cfoutput>
    variable = "variable name">
    the content
</cfoutput>
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

Caching parts of ColdFusion pages in Optimizing ColdFusion applications in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable</td>
<td>Required</td>
<td></td>
<td>Name of the variable in which to save the generated content of the tag.</td>
</tr>
</tbody>
</table>

Usage

This tag requires an end tag. You cannot use this tag to suppress output from a tag library.

Example

The following example uses a custom tag to generate a report and saves the report in the variable CONTENT. It replaces all instances of the word "report" with the phrase "MyCompany Quarterly Report" and outputs the result.
<cfsavecontent variable="content">
    <CF_OutputBigReport>
</cfsavecontent>
<cfoutput>
    #replace(content, "report", "MyCompany Quarterly Report", "all")#
</cfoutput>
cfschedule

Description

Provides a programmatic interface to the ColdFusion scheduling engine. Can run a CFML page at scheduled intervals, with the option to write the page output to a static HTML page. This feature enables you to schedule pages that publish data, such as reports, without waiting while a database transaction is performed to populate the page.

ColdFusion does not invoke Application.cfc methods, when invoking a task's event handler methods.

Category

Variable manipulation tags; Using Scheduler

Syntax

```xml
<cfschedule
  action = "run|update|pause|delete|pauseall|resumeall|list"
  task = "task name"
  endDate = "date"
  endTime = "time"
  file = "filename"
  interval = "seconds"
  operation = "HTTPRequest"
  password = "password"
  path = "path to file"
  port = "port number"
  proxyPassword = "password"
  proxyPort = "port number"
  proxyServer = "host name"
  proxyUser = "user name"
  publish = "yes|no"
  requestTimeOut = "seconds"
  resolveURL = "yes|no"
  isDaily = "yes|no"
  overwrite = "yes|no"
  startDate = "date"
  startTime = "time"
  url = "URL"
  username = "user name">
  group="group1"
  oncomplete="how to handle exception"
  eventhandler="path_to_event_handler"
  onException="refire|pause|invokeHandler"
  cronTime="time"
  repeat="number"
  priority="integer"
  exclude="date|date_range|comma-separated_dates"
  onMisfire = ""
  cluster="yes|no"
  mode="server|application"
  retryCount="number"
</cfschedule>

OR

<cfschedule
  action = "delete"
  task = "task name">
```

© 2014 Adobe Systems Incorporated. All rights reserved.
OR

```cfschedule
  action = "run"
  task = "task name"
```

OR

```cfschedule
  action = "pauseAll"
  mode = "server|application"
```

OR

```cfschedule
  action = "resumeAll"
  mode = "server|application"
```

OR

```cfschedule
  action = "list"
```
mode = "server|application"
result = "res">

⚠️ **Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfcookie, cfparam, cfregistry, cfsavecontent, cfset

History

ColdFusion 10: Added the actions list, pauseall, and resumeall. Also, added the attributes group, onComplete, eventHandler, onException, cronTime, repeat, result, priority, exclude, onMisfire, cluster, mode, isDaily, overwrite, and retryCount.

ColdFusion MX 6.1: Changed the way intervals are calculated. The day length now reflects changes between standard and daylight saving times. The month length is now the calendar month length, not four weeks. The scheduler handles leap years correctly.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| action    | Required|         | • delete: Deletes the specified task.  
• update: Updates an existing task or creates a task, if one with the name specified by the task attribute does not exist.  
• run: Executes the specified task.  
• pause: Pauses the specified task.  
• resume: Continues executing the specified task.  
• list: Lists all the scheduled tasks.  
• pauseAll: Pauses all scheduled tasks for a particular application.  
• resumeAll: Resumes all scheduled tasks for a particular application. |
<table>
<thead>
<tr>
<th><strong>task</strong></th>
<th><strong>Required</strong></th>
<th><strong>Name of the task.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>endDate</strong></td>
<td><strong>Optional</strong></td>
<td><strong>Date when scheduled task ends. The default date format is mm/dd/yy.</strong></td>
</tr>
<tr>
<td><strong>endTime</strong></td>
<td><strong>Optional</strong></td>
<td><strong>Time when scheduled task ends (seconds).</strong></td>
</tr>
<tr>
<td><strong>file</strong></td>
<td><strong>Required if publish = &quot;Yes&quot;</strong></td>
<td><strong>Name of the file in which to store the published output of the scheduled task. The file can only have a .txt or .log extension by default. You can add more extensions in cfusion\lib\neo-cron xml.</strong></td>
</tr>
</tbody>
</table>
| **interval** | | **Interval at which task is scheduled:**
| | | • number of seconds
| | | • once
| | | • daily
| | | • weekly
<p>| | | • monthly |
| <strong>operation</strong> | <strong>Optional</strong> | <strong>HTTPRequest</strong> | <strong>Operation that the scheduler performs.</strong> |
| <strong>overwrite</strong> | <strong>true</strong> | <strong>If false, results in the creation of new output files every time the task executes. If true, instead of creating new outputfiles, the existing one is overwritten.</strong> |
| <strong>password</strong> | <strong>Optional</strong> | <strong>Password, if URL is protected.</strong> |
| <strong>path</strong> | <strong>Required if publish = &quot;Yes&quot;</strong> | <strong>Path to the directory in which to put the published file.</strong> |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>Optional</td>
<td>80</td>
<td>Port to use on the server that is specified by the <code>url</code> parameter. If <code>resolveURL = &quot;yes&quot;</code>, retrieved document URLs that specify a port number are automatically resolved, to preserve links in the retrieved document. A port value in the <code>url</code> attribute overrides this value.</td>
</tr>
<tr>
<td>proxyPassword</td>
<td>Opt</td>
<td></td>
<td>Password to provide to the proxy server.</td>
</tr>
<tr>
<td>proxyPort</td>
<td>Optional</td>
<td>80</td>
<td>Port number to use on the proxy server.</td>
</tr>
<tr>
<td>proxyServer</td>
<td>Optional</td>
<td></td>
<td>Host name or IP address of a proxy server.</td>
</tr>
<tr>
<td>proxyUser</td>
<td>Opt</td>
<td></td>
<td>User name to provide to the proxy server.</td>
</tr>
</tbody>
</table>
| publish    | Optional   | no      | • yes: saves the result to a file.  
• no |
| requestTimeOut | Optional    |         | Can be used to extend the default time-out period. |
| resolveURL | Optional   | no      | • yes: resolves links in the output page to absolute references.  
• no |
<p>| result     | Required for the list action. |         | Name for the query in which <code>cfschedule</code> returns the result variables. |
| startDate  |            |         | Date on which to first run the scheduled task. The default date format is mm/dd/yy. |
| startTime  |            |         | Time at which to run the scheduled task. |
| url        | Required if <code>action = &quot;update&quot;</code> |         | URL of the page to execute. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Optional</td>
<td>User name, if URL is protected.</td>
</tr>
<tr>
<td>group</td>
<td>default</td>
<td>The group to which the scheduled task belongs.</td>
</tr>
<tr>
<td>onComplete</td>
<td>invokeHandler</td>
<td>The action to be performed after the completion of current task. Used when chaining scheduled tasks. For details, see Using Scheduler.</td>
</tr>
<tr>
<td>eventHandler</td>
<td>Optional</td>
<td>A CFC file whose pre-defined methods are invoked for various events while running the task. The CFC must be implementing ITaskEventHandler. The path you specify must be relative to webroot. For example, schedulerdemo.eventhandler.</td>
</tr>
</tbody>
</table>
| onException   | invokeHandler, if eventHandler is specified | Specify what to do if a task results in error. If unspecified, then no action is taken. The options are:  
- **refire:** Tries to run the task immediately.  
- **pause:** Stops the task from executing further.  
- **invokeHandler:** Invokes onError method of the eventhandler as defined by the user. |
| cronTime      | Optional| Specifying task scheduling time in cron job syntax.                           |
| repeat        | -1      | The number of times a task has to be repeated.                               |
| priority      | 5       | An integer that indicates the priority.                                      |
exclude  Optional  Comma-separated list of dates or date range for exclusion in the schedule period.

onMisfire  Optional  invokeHandler if eventHandler is specified  Specify what to do if a task misfires. If unspecified, then no action is taken. The options are:
- fire_now: Tries to run the task immediately.
- invokeHandler: Invokes onMisfire method of the event handler as defined by the user.

cluster  Optional  no  If yes, the task can be executed in cluster setup.

mode  Optional  server  If the task is server-specific or application specific.

retryCount  Optional  3  The number of reattempts if the task fails. The value must be greater than or equal to 0 and less than or equal to 3.

ℹ️ Note: ColdFusion does not invoke Application.cfc methods, when invoking a task's event handler methods.

Usage

This tag and the ColdFusion Administrator Scheduled task page schedule ColdFusion tasks. Tasks that you add or change using this tag are visible in the Administrator. You can disable this tag in the Administrator Sandbox/Resource security page. This tag's success or failure status is written to the schedule.log file in the cf_root/cfusion/logs directory (or the logs directory of another instance you may have created with the ColdFusion Enterprise Instance Manager).

When you create a task, you specify the URL of the ColdFusion page to execute, the date, time and frequency of execution, and whether to publish the task output to an HTML file. If the output is published, you specify the output file path and file.

If you schedule a job to run monthly on any date in the range 28-31, the scheduler does the following:

- If you schedule a monthly job to run on the last day of a month, the scheduled job will run on the last day of each month. For example, if you schedule a monthly job to start on January 31, it will run on January 31, February 28 or 29, March 31, April 30, and so on.
- If you schedule a monthly job to run on the 29th or 30th of the month, the job will run on the specified day of
each month for 30 or 31-day months, and the last day of February. For example, if you schedule a monthly job to start on January 30, the job will run on January 30, February 28 or 29, March 30, April 30, and so on.

If you schedule a job to run once, the starting time is in the past, and the task has not yet run, it runs immediately. If you schedule a recurring job with a start time in the past, ColdFusion schedules the job to run on the next closest interval in the future.

The Scheduler configuration file, `cf_root\lib\neo-cron.xml` contains all scheduled events, as individual entries (except the clustered tasks).

Example

```xml
<h3>cfschedule Example</h3>
<!--- This read-only example schedules a task. To run the example, remove the comments around the code and change the startDate, startTime, url, file, and path attributes to appropriate values. --->

<!---
cfschedule action = "update"
task = "TaskName"
operation = "HTTPRequest"
url = "http://127.0.0.1/playpen/history.cfm"
startDate = "8/17/09"
startTime = "12:25 PM"
interval = "3600"
resolveURL = "Yes"
publish = "Yes"
file = "sample.html"
path = "c:\inetpub\wwwroot\playpen"
requestTimeOut = "600"-->

--->
```
**cfscript**

- **Description**
  Encloses a code block that contains `cfscript` statements.

- **Category**
  Application framework tags, Other tags

- **Syntax**
  ```
  <cfscript>
  cfscript code here
  </cfscript>
  ```

- **See also**
  cfinvoke, cfmessagebox, CreateObject; Extending ColdFusion Pages with CFML Scripting in the Developing ColdFusion Applications

- **History**
  ColdFusion 11: Enhanced support like usage of custom tags.

  ColdFusion MX:
  - Changed how to invoke component methods: this tag can now invoke component methods, using the CreateObject function
  - Changed use of reserved words: you cannot use ColdFusion reserved words within this tag
  - Added the `try` and `catch` statements.

- **Usage**
  Performs processing in CFScript. This tag uses ColdFusion functions, expressions, and operators. You can read and write ColdFusion variables within this tag. For a detailed description of the CFScript scripting language,
including documentation of CFScript statements and the CFScript equivalents of CFML tags, see Extending ColdFusion Pages with CFML Scripting in the Developing ColdFusion Applications.

You can use this tag to enclose a series of assignment statements that would otherwise require `cfset` statements.

If you code a `cftry/cfcatch` block within this tag using an exception's Java class name, provide the fullyqualified class name._

You cannot use some ColdFusion reserved words in this tag. You cannot put a user-defined function whose name begins with any of these strings within this tag:

- `cf`
- `cf_
- `_cf`
- `coldfusion`
- `coldfusion_
- `_coldfusion`

You cannot use the `elseif` construct within a `cfscript` tag. You can use code such as the following:

```csh
cs
else if ( condition )
{
  ...
}
```

Keywords

The following words are now treated as keywords:

- `import`
- `finally`
- `component`
- `interface`
- `pageencoding`

Exception handling with the `cfscript` tag

To handle exceptions with this tag, use `try` and `catch` statements, which are equivalent to the `cftry` and `cfcatch` tags. For each `try` statement, you must have a `catch` statement. In the `catch` block, the variable `exceptionType` contains the exception type. This variable is the equivalent of the `cfcatch` tag built-in variable `cfcatch.Type`. For more information, see Extending ColdFusion Pages with CFML Scripting in the Developing ColdFusion Applications.

Invoking ColdFusion components with the `cfscript` tag

CFScript invokes component methods using the `CreateObject` function. The following example shows how to invoke a component object with the `cfscript` tag, using ordered arguments:
The following example shows how to use an attribute collection within the `cfscript` tag to pass parameters when invoking a component object. An attribute collection is a structure in which each key corresponds to a parameter name and each value is the parameter value passed for the corresponding key.

```cfscript
<cfscript>
stArgs = structNew();
stArgs.zipcode = "55987";
</cfscript>

...<cfinvoke
webservice = "http://www.xmethods.net/sd/2001/TemperatureService.wsdl"
method = "getTemp"
argumentCollection = "#stArgs#"
returnVariable = "aTemp" >
<cfoutput>The temperature at zip code 55987 is #aTemp#</cfoutput>
```

In this example, the structure is created in a `cfscript` block, but you can use any ColdFusion method to create the structure.

Consuming web services with the `cfscript` tag

The following example shows how to consume a web service with the `cfscript` tag. You use the `CreateObject` function to connect to the web service.

```cfscript
<cfscript>
ws = CreateObject("webservice",
    "http://www.xmethods.net/sd/2001/TemperatureService.wsdl");
xlatstring = ws.getTemp("55987");
writeoutput(xlatstring);
</cfscript>
```

For more information, see Using Web Services in the Developing ColdFusion Applications.

Script support for tags

ColdFusion traditionally allowed you to build your applications using simple tags. ColdFusion 11 allows you to write all the tags in the script format in a very generic manner. The format for the generic syntax is as follows:
Note that the attributes of a tag must be enclosed within a parenthesis and also the attributes must be comma-separated.

Consider the following example:

```coldfusion
<cfscript>
  cftag1 (att1=val1, att2=val2) // The parent tag
  {
    // First child tag having attributes in the parenthesis (Optional)
    child_Tag1 (child_attr1='cv1', child_attr2='cv2')
    {
      // Nested child tag
      child_childTag (child_child_attr1=ccv1, child_child_attr2=ccv2);
    }
    // Second child of parent tag
    child_Tag2 (child2_attr1='cv1', child2_attr2='cv2');
  }
</cfscript>
```

You have been used to writing ColdFusion code as mentioned in the above example. To give you more control, invoking tags inside the ColdFusion script block is now supported.

So, now you can write your code in the following manner:

```coldfusion
<cfscript>
  cfform (name='myForm', action='')
  {
    cfgrid (name="myGrid", query='userNameQuery')
    {
      cfgridcolumn (name='firstName', id='firstName', type='string');
      cfgridcolumn (name='lastName', id='LastName', type='string');
    }
  }
</cfscript>
```
The script support is not available for the following tags:

- `<cfscript>`
- `<cfoutput>` (use `writeOutput()` instead)
- `<cfdump>` (use `writeDump()` instead)
- `<cfinvoke>` (use `invoke()` instead)
- `<cfinvokeargument>`
- `<cfobject>` (use `createObject` instead)

### Using custom tags in scripts

Custom tags can also be used in `<cfscript>` blocks just like normal ColdFusion tags. For instance,

If there is a custom tag named `cf_happybirthday` to generate a birthday message which is used in the tag syntax like:

```html
<cf_happybirthday name="Ted Cantor" birthDate="December 5, 1987">
```

The same can be written in the script syntax as:

```html
cf_happybirthday (name="Ted Cantor", birthDate="December 5, 1987");
```

Also, script support for `cfimport`-based prefix custom tag is supported. You can use prefix-based custom tag in script blocks as follows:

```html
<cfscript>
    cfimport(taglib=" ../importFolder/", prefix="myTags");
    myTags:customTag();
</cfscript>
```

In the above example, `customTag` is a CFM file under the `importFolder` directory.

### Example

```html
<p>This simple example shows variable declaration and manipulation.<br>
<cfif IsDefined("form.myValue")>
    <cfif IsNumeric(form.myValue)>
        <cfset x = form.myValue>
        <cfscript>
            y = x;
            z = 2 * y;
            StringVar = form.myString;
        </cfscript>
        <cfoutput>
            <p>twice #x# is #z#.</p>
            Your string value was: <b><I>#StringVar#</i></b>
        </cfoutput>
    </cfif>
</cfif>
```
cfsearch

Description

Searches one or more Solr collections. A collection must be created and indexed before this tag can return search results. A collection can be created in these ways:

- With the cfcollection tag
- In the ColdFusion Administrator
  A collection can be indexed in the following ways:
- In ColdFusion, with the cfindex tag
- In the ColdFusion Administrator, which calls the cfindex tag
  For more information, see Building a Search Interface in the Developing ColdFusion Applications.

Category

Extensibility tags

Syntax

```cfsearch supports script style syntax:
new search().search(name="<search_name>", collection="<collection_name>"});
```

```cfsearch supports script style syntax:
<cfsearch
collection = "collection name"
name = "search name"
category = "category[,category2,...]"
categoryTree = "tree location"
contextBytes = "number of bytes"
contextHighlightBegin = "HTML string"
contextHighlightEnd = "HTML string"
contextPassages = "number of passages"
criteria = "search expression"
language = "language"
maxRows = "number"
orderBy = "rank_order"
previousCriteria = "criteria"
startRow = "row number"
status = ""
suggestions = "suggestion option"
type = "criteria">
```

⚠️ Note

You can specify this tag's attributes in anattributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also
cfcollection, cfexecute, cfindex, cfobject, cfreport, cfwddx

History

ColdFusion 10: New attribute {{orderBy}}

ColdFusion 9: Added support for Solr search engine.

ColdFusion MX 7:

- Added category, categoryTree, status, suggestions, contextPassages, contextBytes, contextHighlightBegin, contextHighlightEnd, and previousCriteria attributes.
- Added author, category, categoryTree, context, rank, size, recordsSearched, and type result columns.
- Added information on the status structure and its associated keys.
- Removed references to a separate K2 server and k2server.ini file.
- Removed references to unregistered collections.
- Removed references to external collections. ColdFusion MX now manages all collections through the Verity Search service.
- Changed cflock recommendation. It is no longer a best practice to surround the cfsearch tag with a cflock tag.

ColdFusion MX:

- Deprecated the external attribute. It might not work, and might cause an error, in later releases. (ColdFusion stores this information about each collection; it automatically detects whether a collection is internal or external.) This tag supports absolute (also known as fully qualified) collection pathnames and mapped collection names.
- Changed query result behavior: the cfindex tag can index the query results from a cfsearch operation.
- Changed Verity operations behavior: ColdFusion supports Verity operations on Acrobat PDF files.
- Changed multiple collection behavior: this tag can search multiple collections. In a multiple collection search, you cannot combine collections that are registered with K2Server and registered in another way.
- Changed acceptable collection naming: this tag accepts collection names that include spaces.
- Changed the following support: this tag supports Verity 2.6.1 and the LinguistX and ICU locales.
- Changed thrown exceptions: this tag can throw the SEARCHENGINE exception.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td>Name of the search query.</td>
<td>Name of the search query.</td>
</tr>
<tr>
<td>collection</td>
<td>Required</td>
<td>One or more collection names. You can specify more than one collection unless you are performing a category search.</td>
<td>One or more collection names. You can specify more than one collection unless you are performing a category search.</td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th><strong>category</strong></th>
<th><strong>Optional</strong></th>
<th>A list of categories, separated by commas, to which the search is limited. If specified, and the collection does not have categories enabled, ColdFusion throws an exception.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>categoryTree</strong></td>
<td><strong>Optional</strong></td>
<td>The location in a hierarchical category tree at which to start the search. ColdFusion searches at and below this level. If specified, and the collection does not have categories enabled, ColdFusion throws an exception. Can be used in addition to the <code>category</code> attribute.</td>
</tr>
<tr>
<td><strong>criteria</strong></td>
<td><strong>Optional</strong></td>
<td>Search criteria. Follows the syntax rules of the <code>type</code> attribute. If you pass a mixed-case entry in this attribute, the search is case sensitive. If you pass all uppercase or all lowercase, the search is case-insensitive. Follow Solr syntax and delimiter character rules; see Solr search support in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td><strong>contextBytes</strong></td>
<td><strong>OptionalSolr</strong></td>
<td>300</td>
</tr>
<tr>
<td><strong>contextHighlightBegin</strong></td>
<td><strong>OptionalSolr</strong></td>
<td><code>&lt;b&gt;</code></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>contextHighlightEnd</td>
<td>OptionalSolr</td>
<td></td>
</tr>
<tr>
<td>contextPassages</td>
<td>OptionalSolr</td>
<td>0</td>
</tr>
<tr>
<td>language</td>
<td>Optional</td>
<td>english</td>
</tr>
<tr>
<td>maxRows</td>
<td>Optional</td>
<td>all</td>
</tr>
<tr>
<td>orderBy</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>previousCriteria</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>startRow</td>
<td>Optional</td>
<td>1</td>
</tr>
<tr>
<td>status</td>
<td>Optional</td>
<td>Specifies the name of the structure variable into which ColdFusion places search information, including alternative criteria suggestions (spelling corrections). For a list of keys in this structure, see the section Status structure keys.</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| suggestions | Optional | never | Specifies whether the search engine returns spelling suggestions for possibly misspelled words. Use one of the following options:  
• Always: Always return spelling suggestions.  
• Never: Never return spelling suggestions.  
• positive integer: Return spelling suggestions if the number of documents retrieved by the search is less than or equal to the number specified. There is a small performance penalty for retrieving suggestion data. |
<table>
<thead>
<tr>
<th>type</th>
<th>Optional</th>
<th>simple</th>
<th>Used to specify the parser that the engine uses to process the criteria.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>simple</td>
<td>• simple: STEM and MANY operators are implicitly used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>explicit</td>
<td>• explicit: operators must be invoked explicitly. Also known as Bool_Plus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>internet</td>
<td>• internet: for documents that are mostly WYSIWIG (what-you-see-is-what-you-get) documents. Also known as Internet_advanced.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>internet_basic</td>
<td>• internet_basic: minimizes search time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>natural</td>
<td>• natural: specifies the Query By Example (QBE parser. Also known as FreeText.</td>
</tr>
</tbody>
</table>

**Usage**

The `cfsearch` tag returns a query object whose columns you can reference in a `cfoutput` tag. For example, the following code specifies a search for the exact terms "filming" or "filmed":

```cfscript
<cfsearch
  name = "mySearch"
  collection = "myCollection"
  criteria = '<WILDCARD>`film{ing,ed}`'
  type="explicit"
  startrow=1
  maxrows = "100">
<cfdump var = "#mySearch#>
```

In this example, the single-quotation mark (') and backtick (`) characters are used as delimiters. To optimize search performance, always specify the `maxrows` attribute, setting it to a value that matches your application's needs. A value less than 300 helps to ensure optimal performance. Adobe does not recommend using the `cflock` tag with this tag; Solr provides the locking function. Using the `{{cflock}}` tag slows search performance.

**The cfsearch tag result columns**

<p>| Variable | Description |
|----------|-------------|-------------|</p>
<table>
<thead>
<tr>
<th>context</th>
<th>A context summary containing the search terms, highlighted in bold (by default). This is enabled if you set the contextpassages attribute to a number greater than zero.</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Value of URLpath attribute in the cfindex tag used to populate a collection.</td>
</tr>
<tr>
<td>key</td>
<td>Value of the key attribute in the cfindex tag used to populate a collection.</td>
</tr>
<tr>
<td>title</td>
<td>Value of title attribute in cfindex tag used to populate the collection, including PDF and Office document titles. If a title is not extracted from the document, the tag uses the cfindex title attribute value for each row.</td>
</tr>
<tr>
<td>score</td>
<td>Relevancy score of document based on search criteria.</td>
</tr>
<tr>
<td>custom1, custom2, custom3, custom4</td>
<td>Value of custom fields in cfindex tag used to populate a collection.</td>
</tr>
<tr>
<td>size</td>
<td>The number of bytes in the index document.</td>
</tr>
<tr>
<td>rank</td>
<td>The rank of this document in the search results.</td>
</tr>
<tr>
<td>author</td>
<td>Extracted from the HTML, Office, and PDF documents when available.</td>
</tr>
<tr>
<td>type</td>
<td>The MIME type of the document.</td>
</tr>
<tr>
<td>category</td>
<td>A list of the categories that were specified for this document when it was indexed.</td>
</tr>
<tr>
<td>categoryTree</td>
<td>A hierarchical category tree, or serial list of categories, that was specified for this document when it was indexed. Only a single tree is returned.</td>
</tr>
<tr>
<td>summary</td>
<td>Contents of automatic summary generated by cfindex.</td>
</tr>
<tr>
<td>recordCount</td>
<td>Number of records returned in record set.</td>
</tr>
<tr>
<td>currentRow</td>
<td>Current row that cfoutput is processing.</td>
</tr>
<tr>
<td>columnList</td>
<td>List of column names within record set.</td>
</tr>
</tbody>
</table>
recordsSearched | Number of records searched. This is the same value for each row in the record set.Corresponds to the search key in the status structure.

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>found</td>
<td>The number of documents that contain the search criteria.</td>
</tr>
<tr>
<td>searched</td>
<td>The number of documents searched. Corresponds to the recordsSearched column in the search results.</td>
</tr>
<tr>
<td>time</td>
<td>The number of milliseconds the search took.</td>
</tr>
<tr>
<td>suggestedQuery</td>
<td>An alternative query that might produce better results. This often contains corrected spellings of search terms. Present only when the suggestions tag attribute criteria is met.</td>
</tr>
<tr>
<td>keywords</td>
<td>A structure containing each search term as a key to an array of up to five possible alternative terms, in order of preference. Present only when the suggestions attribute criteria is met.</td>
</tr>
<tr>
<td>keywordScore</td>
<td>A structure in the same format as for keywords.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<!-- #1 (TYPE=SIMPLE) ------------------------------------------>
<cfsearch
    name="name"
    collection="snippets,syntax,snippets"
    criteria="example"
    maxrows = "100">
<p>
<cfoutput>Search Result total = #name.RecordCount#</cfoutput><br>
<cfoutput>
    url=#name.url#<br>
    key=#name.key#<br>
    title=#name.title#<br>
    score=#name.score#<br>
    custom1=#name.custom1#<br>
    custom2=#name.custom2#<br>
    summary=#name.summary#<br>
    recordcount=#name.recordcount#<br>
    currentrow=#name.currentrow#<br>
    columnlist=#name.columnlist#<br>
    recordssearched=#name.recordssearched#<br>
</cfoutput>
```
<cfdump var = #name#>
<br>

<!---- #2 (TYPE=EXPLICIT) ----------------------------->
<cfsearch>
  name = "snippets"
  collection = "snippets"
  criteria = '<wildcard>`film{ing,ed}`'
  type="explicit"
  startrow=1
  maxrows = "100">
  <cfoutput query="snippets">
    url=#url#<br>
    key=#key#<br>
    title=#title#<br>
    score=#score#<br>
    custom1=#custom1#<br>
    custom2=#custom2#<br>
    summary=#summary#<br>
    recordcount=#recordcount#<br>
    currentrow=#currentrow#<br>
    columnlist=#columnlist#<br>
    recordssearched=#recordssearched#<br>
  </cfoutput>
</cfsearch>
<cfdump var = #snippets#>
<br>

<!---- #3 (search by CF key) ----------------------------->
<cfsearch>
  name = "book"
  collection = "custom_book"
  criteria = "cf_key=bookid2"
  maxrows = "100">
  <cfoutput>
    url=#book.url#<br>
    key=#book.key#<br>
    title=#book.titleE#<br>
    score=#book.score#<br>
    custom1=#book.custom1#<br>
    custom2=#book.custom2#<br>
    summary=#book.summary#<br>
    recordcount=#book.recordcount#<br>
    currentrow=#book.currentrow#<br>
    columnlist=#book.columnlist#<br>
    recordssearched=#book.recordssearched#<br>
  </cfoutput>
</cfsearch>
\[</cfoutput>
\<cfdump var = \#book\>
cfselect

Description

Constructs a drop-down list box form control. Used in a cfform tag. You can populate the list from a query, or by using the HTML option tag.

Category

Forms tags

Syntax

```xml
<cfselect
   name="name"  bind="bind expression"
   bindAttribute="attribute name"
   bindOnLoad="yes|no"
   display="text"  editable="yes|no"
   enabled="yes|no"
   group="query column name"  height="number of pixels"
   id="HTML id"
   label="label"  message="text"  multiple="yes|no"
   onBindError="JavaScript function name"
   onChange="JavaScript or ActionScript"
   onClick="JavaScript function name"
   onError="JavaScript"
   onKeyDown="JavaScript or ActionScript"
   onMouseDown="JavaScript or ActionScript"
   onMouseUp="JavaScript or ActionScript"
   query="query name"  queryPosition="above|below"
   required="yes|no"
   selected="value or list"  size="integer"  sourceForTooltip="URL"
   style="style specification"  tooltip="text"
   value="text"  visible="yes|no"
   width="number of pixels">
   zero or more HTML option tags
</cfselect>
```

Some attributes apply to only specific display formats. For details see the Attributes table.

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfapplet, cfcalendar, cfform, cfformgroup, cfformitem, cfgrid, cfinput, cfslider, cftextarea, cftree; Introduction to Retrieving and Formatting Data and Using Ajax User Interface Components and Features in the Developing ColdFusion Applications

History
ColdFusion 8:

- Added support for binding in HTML format forms, including the `bind`, `bindAttribute`, and `bindOnLoad`, and `onBindError` attributes.
- Added support for tool tips in HTML format forms, including the `sourceForTooltip` attribute.

ColdFusion MX 7:

- When populating a `cfselect` tag with a query when the database field has spaces instead of a value, the error processing of the `cfform` tag required field is not triggered as it was in ColdFusion MX 6.1.
- Added support for specifying multiple values to the `selected` attribute.
- Deprecated the `passthrough` attribute. The tag now supports all HTML `select` tag attributes directly.
- Added `on-prefixed attributes`.
- Added `enabled`, `group`, `height`, `label`, `queryPosition`, `tooltip`, `visible`, and `width` attributes.

Attributes

The following table lists attributes that ColdFusion uses directly. The tag also supports all HTML `select` tag attributes that are not on this list, and passes them directly to the browser.

![Note]

Attributes that are marked as Flash only are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt;Format</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required;All</td>
<td></td>
<td>Name of the select form element.</td>
</tr>
<tr>
<td>bind</td>
<td>Optional; HTML</td>
<td></td>
<td>A bind expression that dynamically sets an attribute of the control. For details, see Usage.</td>
</tr>
<tr>
<td>bindAttribute</td>
<td>Optional;HTML</td>
<td>Populate the options</td>
<td>Specifies the HTML tag attribute whose value is set by the <code>bind</code> attribute. You can only specify attributes in the browser's HTML DOM tree, not ColdFusion-specific attributes. Ignored if there is no <code>bind</code> attribute.</td>
</tr>
<tr>
<td>bindOnLoad</td>
<td>Optional;HTML</td>
<td>no</td>
<td>A Boolean value that specifies whether to execute the bind attribute expression when first loading the form. Ignored if there is no bind attribute.</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>-------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>display</td>
<td>Optional;All</td>
<td>Value of value attribute</td>
<td>Query column to use for the display label of each list element. Used with the query attribute.</td>
</tr>
<tr>
<td>editable</td>
<td>Optional; Flash</td>
<td>no</td>
<td>Boolean value specifying whether you can edit the contents of the control.</td>
</tr>
<tr>
<td>enabled</td>
<td>Optional; Flash</td>
<td>yes</td>
<td>Boolean value specifying whether the control is enabled. A disabled control appears in light gray. The inverse of the disabled attribute.</td>
</tr>
<tr>
<td>group</td>
<td>Optional; HTML and XML</td>
<td></td>
<td>Query column to use to group the items in the drop-down list into a two-level hierarchical list.</td>
</tr>
<tr>
<td>height</td>
<td>Optional; Flash</td>
<td></td>
<td>The height of the control, in pixels.</td>
</tr>
<tr>
<td>id</td>
<td>Optional;HTML</td>
<td>Value of name attribute</td>
<td>The HTML ID of the control.</td>
</tr>
<tr>
<td>label</td>
<td>Optional;Flash and XML</td>
<td></td>
<td>Label to put next to the control on a Flash or XML-format form.</td>
</tr>
<tr>
<td>message</td>
<td>Optional;All</td>
<td></td>
<td>Message to display if required = &quot;yes&quot; and no selection is made.</td>
</tr>
</tbody>
</table>
| multiple            | Optional;All  | no   | • yes: allows selecting multiple elements in drop-down list.  
|                      |               |      | • no |

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Event</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onBindError</td>
<td>Optional; HTML</td>
<td>See Description</td>
<td>The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. (The status code is 1 if the error is not an HTTP error.) If you omit this attribute, and specified a global error handler (by using the <code>ColdFusion.setGlobalErrorHandler</code> function), it displays the error message; otherwise a default error pop-up appears.</td>
</tr>
<tr>
<td>onChange</td>
<td>Optional; All</td>
<td></td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the control changes due to user action.</td>
</tr>
<tr>
<td>onClick</td>
<td>Optional; HTML and XML</td>
<td></td>
<td>JavaScript to run when the user clicks the control.</td>
</tr>
<tr>
<td>onError</td>
<td>Optional; HTML and XML</td>
<td></td>
<td>Custom JavaScript function to execute if validation fails.</td>
</tr>
<tr>
<td>onKeyDown</td>
<td>Optional; All</td>
<td></td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the user presses a keyboard key in the control.</td>
</tr>
<tr>
<td>onKeyUp</td>
<td>Optional; All</td>
<td></td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a keyboard key in the control.</td>
</tr>
<tr>
<td>onMouseDown</td>
<td>Optional; All</td>
<td></td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the user releases a mouse button in the control.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>onMouseUp</td>
<td>Optional;All JavaScript (HTML/XML) or ActionScript (Flash) to run when the user presses a mouse button in the control.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>query</td>
<td>Optional;All Name of query to populate drop-down list.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>queryPosition</td>
<td>Optional;All above If you populate the options list with a query and use HTML option child tags to specify additional entries, this attribute determines the location of the items from the query relative to the items from the option tags:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>required</td>
<td>Optional;All no Note: This attribute has no effect if you omit the size attribute or set it to 1, because the browser always submits the displayed item. You can work around this issue: format forms by having an initial option tag with value=&quot; &quot; (notice the space character between the quotation marks). This workaround applies only for Flash forms and not for HTML forms or XML forms.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **above**: puts the query items above the options items.
- **below**: puts the query items below the options items.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>selected</code></td>
<td>Optional; All</td>
<td>One or more option values to preselect in the selection list. To specify multiple values, use a comma-delimited list. This attribute applies only if selection list items are generated from a query. The <code>cfform</code> <code>preserveData</code> attribute can override this value.</td>
</tr>
<tr>
<td><code>size</code></td>
<td>Optional; All</td>
<td>1 Number of entries to display at one time. The default value displays a drop-down list. Any other value displays a list box with <code>size</code> number of entries visible at one time.</td>
</tr>
<tr>
<td><code>sourceForTooltip</code></td>
<td>Optional; HTML</td>
<td>The URL of a page to display as a tool tip. The page can include CFML and HTML markup to control the tip contents and format, and the tip can include images. If you specify this attribute, an animated icon appears with the text &quot;Loading...&quot; while the tip is being loaded.</td>
</tr>
<tr>
<td><code>style</code></td>
<td>Optional; All</td>
<td>In HTML or XML format forms, ColdFusion passes the <code>style</code> attribute to the browser or XML. In Flash format, must be a style specification in CSS format, with the same syntax and contents as used in Flex for the corresponding Flash element.</td>
</tr>
</tbody>
</table>
tooltip | Optional; Flash, HTML | Text to display when the mouse pointer hovers over the control. The text can include HTML markup. Ignored if you specify a sourceForTooltip attribute.

value | Optional; All | Query column to use for the value of each list element. Used with the query attribute.

visible | Optional; Flash | yes | Boolean value that specifies whether to show the control. The display reserves empty space for an invisible control.

width | Optional; Flash | The width of the control, in pixels.

Usage

For this tag to work properly, the browser must have JavaScript enabled. This tag requires an end tag and can include HTML option and optgroup child tags. To ensure that a selected list box item persists across postbacks, use the cfform tag preserveData attribute with a list generated from a query. (This strategy works only with data that is populated from a query.) If the cfform preserveData attribute is yes and the form posts back to the same page, and if the control is populated by a query, the posted selections for the cfselect control are used instead of the Selected attribute. For controls that are populated with regular HTML option tags, the developer must dynamically add the Selected attribute to the appropriate option tags.

The group option generates a query by using SQL GROUP BY syntax and places the value column entries from each group in an indented list under the group column's field value. This option generates an HTML optgroup tag for each entry in the group column. Close each HTML option tag in the cfselect tag body with a </option> end tag. If you do not do so, and you specify queryPosition="below", the first item from the query might not appear in the list.

The bind attribute lets you set cfselect attributes dynamically. Often, it is used to dynamically create the options list based on values that the user enters in the form. For example, you can use the bind attribute to create a Cities option list based on the user's selection from a States cfselect control. When you use a bind attribute to populate the selection list, the function or URL that returns the selection values must return one of the following:

- A two-dimensional array, where the first element in each array row is the option value and the second element in the row is the text to display in the option list.
- If the bind specifies a CFC function, a query, or, if the bind specifies a URL, a JSON representation of a query. The query must include columns whose names are the values of the cfselect tag value and display attributes. Although you can return additional columns, you cannot use them in your client-side code. When you call a CFC function, you do not have to convert the query to JSON format yourself; ColdFusion automatically does the conversion.

To use this format, specify a value attribute. If you omit the display attribute, you must have only a single column in the query that contains the values; the values are also used as the displayed text.
For detailed information on binding, see Binding data to form fields in the Developing ColdFusion Applications. For more information, see the cfform tag entry.

Example

Example 1: Without data binding
The following example lets you select one or more employee names from a list of all employees, grouped by departments, and displays the selected names and the employee’s e-mail addresses. It includes an option to get data for all employees.

```coldfusion
<!--- Get the employee names from the database. --->
<cfquery name = "GetAllEmployees" dataSource = "cfdocexamples"
cachedwithin="#createTimeSpan(0,1,0,0)#">
SELECT Emp_ID, EMail, Phone, Department, FirstName, LastName,
FirstName || ' ' || lastName as Name
FROM Employees
GROUP BY Department, Emp_ID, EMail, Phone, FirstName, LastName, FirstName
</cfquery>

<h2>cfselect Example</h2>
<!-- The cfif statement is true if the form was submitted. Show the selected names. --->
<cfif IsDefined("form.employeeid")>
<!--- The form was submitted. --->
<h3>You Selected the following employees</h3>
<cfif form.employeeid IS ">
<!--- Select All option was selected. Show all employees. --->
<cfoutput query="GetAllEmployees">
#name#<br>
Email: #email#<br><br>
</cfoutput>
<cfelse>
<!--- Use a query of queries to get the data for the selected users. Form.employeeid is a comma-delimited list of selected employee IDs. --->
<cfquery name = "GetSelectedEmployees" dbtype="query">
SELECT Emp_ID, EMail, Phone, Department, FirstName, LastName,
FirstName || ' ' || lastName as Name
FROM GetAllEmployees
WHERE Emp_ID in (#form.employeeid#)
</cfquery>
<cfoutput query="GetSelectedEmployees">
#firstName# #lastName#<br>
Email: #email#<br>
<br>
</cfoutput>
</cfif>
</cfif>

<!--- The cfform tag posts back to the current page. --->"
<cfform action="#CGI.SCRIPT_NAME#">
  <!---
  Use cfselect to present the query's LastName column,
  grouped by department.
  Allow Multiple selections.
  --->
  <cfselect
      name = "employeeid"
      size = "15"
      multiple="yes"
      required = "Yes"
      message = "Select one or more employee names"
      query = "GetAllEmployees"
      group="Department"
      display ="name"
      value ="emp_id"
      queryPosition="Below">
    <option value = ">Select All</option>
  </cfselect>
</cfform>
Example 2: With data binding
The following example uses binding to fill in the options list of the Cities control only after the user selects a state. (In this example, only two states, California and New Jersey, have city entries.) The CFML page is the simplest part of the example. It consists of the following lines:

```html
<html>
<head>
</head>
<body>
<cfform name="mycfform">
<!---
The States selector. The bindonload attribute is required to fill the selector.--->
<cfselect name="state" bind="cfc:bindFcns.getstates()" bindonload="true">
  <option name="0">--state--</option>
</cfselect>
<cfselect name="city" bind="cfc:bindFcns.getcities({state})">
  <option name="0">--city--</option>
</cfselect>
</cfform>
</body>
</html>
```

The BinFcns CFC has three functions: getstates, to get the states; getcities, to get the cities; and an internal getXmlData function that parses an XML file to get the state and city information. The following examples shows the CFC:
<cfcomponent>
    <cffunction name="getXmlData" output="true">
        <cfset var xmlData = ">
        <cffile action="read" file="#expandpath('.')#\states.xml" variable="xmlData">
            <cfset xmlData = XmlParse(xmlData)>
            <cfreturn xmlData></cffunction>

    <cffunction name="getstates" access="remote">
        <cfset state = arraynew(2)>
        <cfset xmlData = getXmlData()>
        <cfset numStates = 0>
        <cfset state[1][1] = "0">
        <cfset state[1][2] = "--state--">
        <cfset numStates = ArrayLen(xmlData.states.XmlChildren)>
        <cfloop from="1" to="#numStates#" index="j">
            <cfset state[j+1][1] = ltrim(xmlData.states.state[j].XmlAttributes.abr)>
            <cfset state[j+1][2] = ltrim(xmlData.states.state[j].name.xmlText)>
        </cfloop>
        <cfreturn state></cffunction>

    <cffunction name="getcities" access="remote">
        <cfargument name="state" required="yes">
        <cfset var city = arraynew(2)>
        <cfset var xmlData = getXmlData()>
        <cfset var numStates = 0>
        <cfset var numCities = 0>
        <cflog text="In getcities">
        <cfset city[1][1] = "0">
        <cfset city[1][2] = "--city--">
        <cftry>
            <cfset numStates = ArrayLen(xmlData.states.XmlChildren)>
            <cfloop from="1" to="#numStates#" index="j">
                <cfif xmlData.states.state[j].XmlAttributes.abr eq state>
                    <cfset numCities = ArrayLen(xmlData.states.state[j].cities.XmlChildren)>
                    <cfloop from="1" to="#numCities#" index="k">
                        <cfset city[k+1][1] = ltrim(xmlData.states.state[j].cities.city[k].XmlAttributes.name)>
                        <cfset city[k+1][2] = ltrim(xmlData.states.state[j].cities.city[k].XmlAttributes.name)>
                    </cfloop>
                    <cfbreak>
                </cfif>
            </cfloop>
        </cftry>
        <cfcatch type="any">
            <!--- Do nothing. --->
        </cfcatch>
        <cfreturn city></cffunction>
</cfcomponent>
The states.xml file has the following code. To keep the code short, only two states have cities, and only four states are listed.

```
<states>
  <state abr="NJ">
    <name>New Jersey</name>
    <cities>
      <city name="Edison" />
      <city name="Rahway" />
      <city name="Atlantic City" />
      <city name="Hoboken" />
      <city name="Jersey City" />
      <city name="Newark" />
      <city name="Trenton" />
      <city name="Union City" />
    </cities>
  </state>
  <state abr="CA">
    <name>California</name>
    <cities>
      <city name="Anaheim" />
      <city name="Beverly Hills" />
      <city name="Elk Grove" />
      <city name="Fairfield" />
      <city name="Fremont" />
      <city name="Indian Wells" />
      <city name="Long Beach" />
    </cities>
  </state>
  <state abr="ME">
    <name>Maine</name>
  </state>
  <state abr="MA">
    <name>Massachusetts</name>
  </state>
</states>
```
cfservlet

Description

This tag is deprecated. Executes a Java servlet on a JRun engine. To access servlets that run on the same server as ColdFusion, use code such as the following, in which `path` specifies a servlet, JSP, or anything else:

```java
getPageContext().include(path)
getPageContext().forward(path)
```

For more information, see the JSP PageContext API or the Servlet RequestDispatcher API.

History

ColdFusion MX: Deprecated this tag. It might not work, and it might cause an error, in later releases.
**cfservletparam**

**Description**

This tag is deprecated. A child tag of the `cfservlet` tag. Passes data to a servlet. Each `cfservletparam` tag within the `cfservlet` block passes a separate item of data to the servlet. To access servlets that run on the same server as ColdFusion, use code such as the following, in which `path` specifies a servlet, JSP, or anything else:

```
getPageContext().include(path)
getPageContext().forward(path)
```

For more information, see the JSP PageContext API or the Servlet RequestDispatcher API.

**History**

ColdFusion MX: Deprecated this tag. It might not work, and it might cause an error, in later releases.
**cfset**

**Description**

Sets a value in ColdFusion. Used to create a variable, if it does not exist, and assign it a value. Also used to call functions.

**Category**

**Variable manipulation tags**

**Syntax**

```html
<cfset var variable_name = expression>
```

**See also**

cfcookie, cfparam, cfregistry, cfsavecontent, cschedule; Elements of CFML in the Developing ColdFusion Applications

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable_name</td>
<td>Required</td>
<td></td>
<td>A variable.</td>
</tr>
<tr>
<td>var</td>
<td>Optional</td>
<td></td>
<td>A keyword. Does not take a value. Identifies the variable as being local to a function. The variable only exists for the time of the current invocation of the function.</td>
</tr>
</tbody>
</table>

**Usage**

You use the `cfset` tag in several ways in your applications.

**Calling functions**

When you use the `cfset` tag to call a function, you do not have to assign the function return value to a variable if the function does not return a value or you do not have to use the value returned by the function. For example, the following line is a valid ColdFusion `cfset` tag for deleting the `MyVariable` variable from the Application scope:

```html
<cfset StructDelete(Application, "MyVariable")>
```

**Arrays**

The following example assigns a new array to the variable `months`:
<cfset months = ArrayNew(1)>

This example creates a variable `Array_Length` that resolves to the length of the `Scores` array:

<cfset Array_Length = ArrayLen(Scores)>

This example assigns, to index position two in the array `months`, the value `February`:

<cfset months[2] = "February">

Dynamic variable names

In this example, the variable name is itself a variable:

<cfset myvariable = "current_value">
<cfset "#myvariable#" = 5>

Function local variables

The `var` keyword specifies that the variable being defined is only available inside the body of a function that you define by using the `cffunction` tag. The variable value that is set in one invocation of the function is not available in any other invocation of the function. The `var` keyword is the equivalent of the `var` statement in CFScript. The following rules apply to the `var` keyword:

- Any `cfset` tag that uses the `var` keyword must be inside the body of a `cffunction` tag. If you use the `var` keyword in a `cfset` tag outside a `cffunction` tag body, ColdFusion displays an error message.
- Place all `cfset` tags that use the `var` keyword at the beginning of the `cffunction` tag body, before any other ColdFusion tags.

The following example shows how to use the new keyword:

<cffunction name="myFunct">
  <cfset var myVar = "This is a test">
  <cfreturn myVar & " Message.">
</cffunction>
<cfoutput>#myFunct()#</cfoutput>

In this example, the variable `myVar` exists only when the function `myFunct` executes, and it is not available elsewhere on the ColdFusion page.

COM objects

In this example, a COM object is created. A `cfset` tag defines a value for each method or property in the COM
object interface. The last `cfset` creates a variable to store the return value from the COM object's `SendMail` method.

```
<cfobject action = "Create"
  name = "Mailer"
  class = "SMTPsvg.Mailer">

  <cfset MAILER.FromName = form.fromname>
  <cfset MAILER.RemoteHost = RemoteHost>
  <cfset MAILER.FromAddress = form.fromemail>
  <cfset MAILER.AddRecipient("form.fromname", "form.fromemail")>
  <cfset MAILER.Subject = "Testing cfobject">
  <cfset MAILER.BodyText = "form.msgbody">
  <cfset Mailer.SMTPLog = "logfile">
  <cfset success = MAILER.SendMail()>
  <cfoutput> #success# </cfoutput>
```

Example
<!--- This example shows how to use cfset. --->
<cfquery name = "GetMessages" dataSource = "cfdocexamples">
    SELECT *
    FROM Messages
</cfquery>

<h3>cfset Example</h3>
<p>cfset sets and reassigns values to local or global variables within a page.
</p>
<cfset NumRecords = GetMessages.recordCount>
<p>For example, the variable NumRecords has been declared on this page to hold the number of records returned from query</p>
(cfoutput><#NumRecords#</cfoutput>).

<p>In addition, cfset can be used to pass variables from other pages, such as this example, which takes the url parameter Test from this link: <a href = "cfset.cfm?test = <cfoutput>
    #URLEncodedFormat("hey, you, get off of my cloud")#
    "click here</A>) to display a message:
</p>
<cfif IsDefined("url.test") is "True">
    <cfoutput><b><i>#url.test#</i></b></cfoutput>
</cfif>
<p>The variable url.test has not been passed from another page.</p>

<p>cfset can also be used to collect environmental variables, such as the time, the IP address of the user, or another function or expression.</p>
<cfset the_date = #DateFormat(Now())# & " " & #TimeFormat(Now())#>
<cfset user_ip = CGI.REMOTE_ADDR>
<cfset complex_expr = (23 MOD 12) * 3>
<cfset str_example = Reverse(Left(GetMessages.body, 35))>

<cfoutput>
<ul>
    <li>The date: #the_date#
    <li>User IP Address: #user_ip#
    <li>Complex Expression ((23 MOD 12) * 3): #complex_expr#
    <li>String Manipulation (the first 35 characters of the body of the first message in our query)
        <br>Reversed</li>: #str_example#
        <br>Normal</li>: #Reverse(str_example)#
</ul>
</cfoutput>
**cfsetting**

**Description**

Controls aspects of page processing, such as the output of HTML code in pages. The cfsetting tag can also be used in script style.

**Category**

Page processing tags, Variable manipulation tags

**Syntax**

```
<cfsetting
    enableCFoutputOnly = "yes|no"
    requestTimeOut = "value in seconds"
    showDebugOutput = "yes|no" >
```

Script style syntax for cfsetting:

```
setting enablecfoutputonly="true" requesttimeout="0" showdebugoutput="yes";
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

cfcache, cfflush, cfheader, cfhtmlhead, cfinclude, cfprocessingdirective, cfsilent; Controlling debugging output with the cfsetting tag in the Developing ColdFusion Applications

**History**

ColdFusion MX 6.1: Changed behavior: if the tag has a body, ColdFusion executes its contents.

ColdFusion MX:

- Added the requestTimeOut attribute.
- The catchExceptionsByPattern attribute is obsolete. It does not work, and causes an error, in releases later than ColdFusion 5.
- Changed exception handling: the structured exception manager searches for the best-fit cfcatch handler. (In earlier releases, an exception was handled by the first cfcatch block that could handle an exception of its type.)

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enableCFoutputOnly</td>
<td>Req/Opt</td>
<td>yes</td>
<td>Controls whether ColdFusion outputs HTML code.</td>
</tr>
<tr>
<td>requestTimeOut</td>
<td>Req/Opt</td>
<td></td>
<td>Controls how long ColdFusion waits for a request to complete.</td>
</tr>
<tr>
<td>showDebugOutput</td>
<td>Req/Opt</td>
<td>yes</td>
<td>Controls whether debugging information is displayed.</td>
</tr>
<tr>
<td>enableCFoutputOnly</td>
<td>Optional</td>
<td>yes</td>
<td>blocks output of HTML that is outside <code>cf output</code> tags.</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
<td>-----</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>requestTimeout</td>
<td>Optional</td>
<td>yes</td>
<td>integer; number of seconds. Time limit, after which ColdFusion processes the page as an unresponsive thread. Overrides the time-out set in the ColdFusion Administrator. If you specify 0 as the value, timeout is disabled for the request.</td>
</tr>
<tr>
<td>showDebugOutput</td>
<td>Optional</td>
<td>yes</td>
<td>yes: if debugging is enabled in the Administrator, displays debugging information.</td>
</tr>
</tbody>
</table>

### Usage

The `cfsetting requestTimeout` attribute replaces the use of `requestTimeOut` within a URL. To enforce a page time-out, detect the URL variable and use code such as the following to change the page time-out:

```cfsetting RequestTimeout = "#URL.RequestTimeout#"```

You can use this tag to manage whitespace in ColdFusion output pages. If you nest `cfsetting` tags: to make HTML output visible, match each `enableCFoutputOnly = "Yes"` statement with an `enableCFoutputOnly = "No"` statement. For example, after five `enableCFoutputOnly = "Yes"` statements, to enable HTML output, you must have five corresponding `enableCFoutputOnly = "No"` statements. If HTML output is enabled (no matter how many `enableCFoutputOnly = "No"` statements have been processed) the first `enableCFoutputOnly = "Yes"` statement blocks output.

If the debugging service is enabled and `showDebugOutput = "Yes"`, the `IsDebugMode` function returns `Yes`; otherwise, `No`.  

© 2014 Adobe Systems Incorporated. All rights reserved.
Note

Releases after ColdFusion MX allow a `<cfsetting>` end tag; however, this end tag does not affect processing. The `cfsetting` attributes affect code inside and outside the `cfsetting` tag body. ColdFusion MX ignored code between `cfsetting` start and end tags.

Example

```
<p>CFSETTING is used to control the output of HTML code in ColdFusion pages. This tag can be used to minimize the amount of generated whitespace.

<cfsetting enableCFoutputOnly = "Yes">
This text is not shown
<cfsetting enableCFoutputOnly = "No">
<p>This text is shown
<cfsetting enableCFoutputOnly = "Yes">
<cfoutput>
<p>Text within cfoutput is always shown
</cfoutput>
<cfsetting enableCFoutputOnly = "No">
<cfoutput>
<p>Text within cfoutput is always shown
</cfoutput>
```
**cfsharepoint**

**Description**

Invokes a feature that SharePoint exposes as a web service action, such as the Document Workspace `getdwsdata` action.

**Category**

*Extensibility tags, MS Office Integration tags.*

**Syntax**

```
<cfsharepoint
    action="webservice action"
    params="parameter structure"
    domain="domain name"
    name="result variable name"
    password="connection password"
    userName="user ID"
    wsdl="WSDL file path">
```

or

```
<cfsharepoint
    action="webservice action"
    params="parameter structure"
    login = "credentials structure"
    name="result variable name"
    wsdl="WSDL file path">
```

**Note**

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

**History**

ColdFusion 9: Added tag.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Required</td>
<td></td>
<td>The name of a web service action. See Usage for the list of service actions you can specify.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Optional</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>domain</td>
<td>Optional</td>
<td>The domain name required to connect to the SharePoint server. Required if you do not specify a login attribute.</td>
<td></td>
</tr>
<tr>
<td>login</td>
<td>Optional</td>
<td>A structure containing user, password, and domain login credentials to pass to the service. If you do not specify domain, password, and userName attributes, specify a login structure with domain, password, and userName entries.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Optional</td>
<td>Name of the result variable in which to put the data returned by the SharePoint service.</td>
<td></td>
</tr>
<tr>
<td>params</td>
<td>Optional</td>
<td>A structure containing names and values of the parameters to pass to the service. This attribute is required for any service that requires parameters.</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td>The password required to connect to the SharePoint server. Required if you do not specify a login attribute.</td>
<td></td>
</tr>
<tr>
<td>userName</td>
<td>Optional</td>
<td>The user name required to connect to the SharePoint server. Required if you do not specify a login attribute.</td>
<td></td>
</tr>
<tr>
<td>wsdl</td>
<td>Optional</td>
<td>Path to the service wsdl file. Required to invoke an action that is not in the list of supported actions. See Usage for details.</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**
The cfsharepoint tag invokes a Microsoft SharePoint web service. You call many SharePoint web service actions by specifying the action name in the action attribute and passing the web service parameters in the params attribute. You access the services and methods that the cfsharepoint tag does not support directly by specifying the service WSDL URLs in the wsdl attribute.

Note
You can use the cfsharepoint tag with servers that use basic authentication only.

You request a service and action by specifying the action attribute values listed in the following tables. In nearly all cases, these are identical to the SharePoint action names. Notes indicate where the attribute values differ from the action names because multiple services have the same action name.

Note

When the cfsharepoint tag receives the results from the SharePoint server and completes, the structure specified by the name attribute contains the response. This structure also has a ResultFlag entry containing the value Success or Failure. The entry value is Success if there is no Axis Fault or an error is returned in the response, otherwise, the value is Failure.

### Document Workspace

<table>
<thead>
<tr>
<th>cancreatedwsurl</th>
<th>deletedwsfolder</th>
<th>renamedws</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdws</td>
<td>finddwsdoc</td>
<td>updatedwsdata</td>
</tr>
<tr>
<td>createdwsfolder</td>
<td>getdwsdata</td>
<td></td>
</tr>
<tr>
<td>deletedws</td>
<td>removedwsuser</td>
<td></td>
</tr>
</tbody>
</table>

Note
The createdwsfolder and deletedwsfolder action attribute values correspond to the createfolder and deletefolder actions of the Document Workspace service.

### Imaging

<table>
<thead>
<tr>
<th>delete</th>
<th>getitemsbyids</th>
<th>upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>download</td>
<td>listpicturelibrary</td>
<td></td>
</tr>
<tr>
<td>getimaginglistitems</td>
<td>rename</td>
<td></td>
</tr>
</tbody>
</table>
### Lists

<table>
<thead>
<tr>
<th>Method</th>
<th>Method</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>addattachment</td>
<td>getattachmentcollection</td>
<td>updatelist</td>
</tr>
<tr>
<td>addlist</td>
<td>getlist</td>
<td>updatelistitems</td>
</tr>
<tr>
<td>deleteattachment</td>
<td>getlistcollection</td>
<td></td>
</tr>
<tr>
<td>deletelist</td>
<td>getlistitems</td>
<td></td>
</tr>
</tbody>
</table>

### Search or spsearch

**Note**

spsearch/search is not present in Windows Sharepoint Services 2.0.

In Windows SharePoint Services 3.0, if the action attribute specifies any of the following actions, the spsearch.asmx web service is used to perform the search. In Microsoft Office SharePoint Portal Server 2003 or Microsoft Office SharePoint Server 2007, search.asmx is used to perform the search. In Windows Sharepoint Services 2.0, an exception is thrown.

<table>
<thead>
<tr>
<th>Method</th>
<th>Method</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>registration</td>
<td></td>
</tr>
<tr>
<td>queryex</td>
<td>status</td>
<td></td>
</tr>
</tbody>
</table>

### UserGroup

<table>
<thead>
<tr>
<th>Method</th>
<th>Method</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>addgrouptorole</td>
<td>getgroupcollection</td>
<td>removerole</td>
</tr>
<tr>
<td>addrole</td>
<td>getrolecollection</td>
<td>removeusercollectionfromgroup</td>
</tr>
<tr>
<td>addusercollectiontogroup</td>
<td>getusercollectionfromrole</td>
<td>removeuserfromgroup</td>
</tr>
<tr>
<td>addusercollectiontorage</td>
<td>getusercollectionfromrole</td>
<td></td>
</tr>
<tr>
<td>addusertogroup</td>
<td>getuserinfo</td>
<td></td>
</tr>
</tbody>
</table>

### Views

<table>
<thead>
<tr>
<th>Method</th>
<th>Method</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>addview</td>
<td>getview</td>
<td>updateview</td>
</tr>
<tr>
<td>deleteview</td>
<td>getviewcollection</td>
<td></td>
</tr>
</tbody>
</table>
Data type conversion

Some web service actions require parameters in a Microsoft data type that does not correspond directly to a ColdFusion data type. The `cfsharepoint` tag automatically converts between the Microsoft data types and the most appropriate Java data types, which ColdFusion uses internally. The following table lists the conversions, and indicates the corresponding ColdFusion data type.

<table>
<thead>
<tr>
<th>SharePoint data type</th>
<th>ColdFusion Java data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>XmlNode</td>
<td>XMLNodeList - corresponds to a ColdFusion XML object, for example created by passing an XML string to the XmlParse function.</td>
</tr>
<tr>
<td>ArrayOfString</td>
<td>string array - corresponds to a ColdFusion array containing string data.</td>
</tr>
<tr>
<td>UnsignedInt</td>
<td>int - corresponds to a ColdFusion number that is an integer value</td>
</tr>
<tr>
<td>ArrayOfUnsignedInt</td>
<td>int array - corresponds to a ColdFusion array containing string data.</td>
</tr>
</tbody>
</table>

Example

The following example shows how you can manipulate lists and views. It requires resources on the SharePoint server that are not specified here.

```xml
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <title>cfsharepoint Views Example</title>
</head>

<body>
  <cfoutput>
    Getting the list collection<br />
    <!--- All login information is defined using variables in the Application.cfc file. --->
    <cfsharepoint action="getlistcollection" login="#login#" name="result"/>
    
    result.ResultFlag: #result.ResultFlag#<br><br>
    Deleting mycustomlist from the collection, if it exists.<br>
    <cfloop array=#result.lists# index="list">
      <cfif list.Title EQ "mycustomlist">
        <cfsharepoint action="deletelist" login="#login#" name="result1" params="#{listname="mycustomlist"}#"/>
      </cfif>
    </cfloop>
    
    Was anything deleted?
    <cfif IsDefined("result1")>

```
YES. The result is:
<cfdump var="#result1#">
<cfelse>
NO</cfif>

Adding a mycustomlist list
<cfsharepoint action="addlist" login="#login#" name="result1"
params="#{listname="mycustomlist", description="Adding a list via cfsharepoint",
templateid=100}#"/>

addlist result.ResultFlag: #result1.ResultFlag#

<cfset viewFields = xmlparse("<ViewFields>
  <FieldRef Name='Title'/>
  <FieldRef Name='ID'/>
</ViewFields>")>

<cfset query = xmlparse("<Query>
  <Where>
    <Lt>
      <FieldRef Name='ID'/>
      <Value Type='Counter'>10</Value>
    </Lt>
  </Where>
  <OrderBy>
    <FieldRef Name='ID'/>
  </OrderBy>
</Query>")>

<cfset rowlimit = xmlparse("<RowLimit Paged='True'>50</RowLimit>" gren

Adding a myview1 view for the mycustomlist list
<cfsharepoint action="addview" login="#login#" name="result"
params="#{listName="mycustomlist", viewname="myview1",
viewFields="#viewFields#", query="#query#", rowlimit="#rowlimit#",
type="grid", makeViewDefault=false}#"/>

addview result.ResultFlag: #result.ResultFlag#

Adding a myview3 view for the mycustomlist list
<cfsharepoint action="addview" login="#login#" name="result"
params="#{listName="mycustomlist", viewname="myview3", viewFields="#viewFields#", query="#query#", rowlimit="#rowlimit#", type="grid", makeViewDefault=false}#"/>

addview result.ResultFlag: #result.ResultFlag#

Getting the updated mycustomlist view collection
<cfsharepoint action="getviewcollection" login="#login#" name="result"
params="#{listName="mycustomlist"}#"/>

<cfdump var="#result#"
The names of the collection's views:
<cfloop array="#result.views#" index=v>
<cfoutput>#v.displayname##</cfoutput>
</cfloop>
<br>

Deleting the list<br>
<cfsharepoint action="deletelist" login="#login#" name="result1"
params="#{listname="mycustomlist"}#"/>

deletelist result.ResultFlag: #result1.ResultFlag#
cfsilent

Description

Suppresses output produced by CFML within a tag’s scope.

Category

Data output tags, Page processing tags

Syntax

```html
<cfsilent>
...
</cfsilent>
```

See also

cfcache, cfflush, cfheader, cfhtmlhead, cfinclude, cfsetting; Writing and Calling User-Defined Functions in the Developing ColdFusion Applications

Usage

This tag requires an end tag.

Example

```html
<h3>cfsilent</h3>

<cfsilent>
<cfset a = 100>
<cfset b = 99>
<cfset c = b-a>
<cfoutput>Inside cfsilent block<br>
b-a = #c#</cfoutput><br>
</cfsilent>

<p>Even information within cfoutput tags does not display within a cfsilent block.<br>
<cfoutput>
b-a = #c#
</cfoutput>
</p>
```
cfslider

Description

Places a slider control, for selecting a numeric value from a range, in a ColdFusion form. The slider moves over the slider groove. As the user moves the slider, the current value displays. Used within a cfform tag for forms in HTML and applet format. Not supported with Flash forms. In HTML forms, you can create visually rich sliders that let you modify adjustable values in fixed increments. You can specify maximum, minimum, and increment values, to help you quickly filter complex results. The sliders are categorized based on the slider control. The available slider controls are:

- **Vertical** The slider has vertical controls that can be adjusted to the top or bottom.
- **Horizontal** The slider has horizontal controls that can be adjusted to the left or right.
- **Tip** The slider displays the values as data tips.
- **Snapping** The slider moves in incremental values.

Category

**Forms tags**

Syntax

For HTML

```html
<cfslider
  name = "name"
  clickToChange = "true|false"
  format = "html"
  height = "integer"
  increment = "Unit increment value"
  max = "maximum value for the slider"
  min = "minimum value for the slider"
  onChange = "JavaScript function name"
  onDrag = "JavaScript function name"
  tip = "true|false"
  value = "integer"
  vertical = "true|false"
  width = "integer">
</cfslider>
```

Syntax

For Applet
<cfslider
    name = "name"
    align = "top|left|bottom|baseline|texttop|absbottom|middle|absmiddle|right"
    bgColor = "color"
    bold = "yes|no"
    font = "font name"
    fontSize = "integer"
    height = "integer"
    hSpace = "integer"
    italic = "yes|no"
    label = "text"
    lookAndFeel = "motif|windows|metal"
    message = "text"
    notSupported = "text"
    onError = "text"
    onValidate = "script name"
    range = "minimum value, maximum value"
    scale = "integer"
    textColor = "color"
    value = "integer"
    vertical = "yes|no"
    vSpace = "integer"
    width = "integer">

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also
cfapplet, cfcalendar, cfform, cfformgroup, cfformitem, cfgrid, cfinput, cfselect, cftextarea, cf tree: Introduction to Retrieving and Formatting Data and Building Dynamic Forms with cfform Tags in the Developing ColdFusion Applications

History

ColdFusion MX: Deprecated the img, imgStyle, grooveColor, refreshLabel, tickmarklabels, tickmark major, tickmarkminor, and tickmarkimages attributes. They sometimes do not work, and can cause an error, in later releases.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td></td>
<td>Name of cfslider control.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>align</td>
<td>Optional</td>
<td></td>
<td>Alignment of slider:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- top</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- left</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- bottom</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- baseline</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- texttop</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- absbottom</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- middle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- absmiddle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- right</td>
</tr>
<tr>
<td>bgColor</td>
<td>Optional</td>
<td></td>
<td>Background color of slider label. For a hexadecimal value, use the form:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bgColor = &quot;##xxxxxx&quot;, where x = 0-9 or A-F; use two number signs or none.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- any color, in hexadecimal format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- black</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- red</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- blue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- magenta</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- cyan</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- orange</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- darkgray</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- pink</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- gray</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- white</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- lightgray</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yellow</td>
</tr>
<tr>
<td>bold</td>
<td>Optional</td>
<td>no</td>
<td>Whether clicking the slider changes the value of the pointer:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- yes: label text in bold</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- no: medium text</td>
</tr>
<tr>
<td>clickToChange</td>
<td>OptionalHTML</td>
<td></td>
<td>Whether clicking the slider changes the value of the pointer:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- false</td>
</tr>
<tr>
<td>font</td>
<td>Optional</td>
<td></td>
<td>Font name for label text</td>
</tr>
<tr>
<td>fontSize</td>
<td>Optional</td>
<td></td>
<td>Font size for label text, in points.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>format</td>
<td>Optional</td>
<td>applet</td>
<td>Specifies if the format is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• html</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• applet</td>
</tr>
<tr>
<td>height</td>
<td>Optional</td>
<td>40, for applet 100, for HTML</td>
<td>Slider control height, in pixels.</td>
</tr>
<tr>
<td>hSpace</td>
<td>Optional</td>
<td></td>
<td>Horizontal spacing to left and right of slider, in pixels.</td>
</tr>
<tr>
<td>italic</td>
<td>Optional</td>
<td>no</td>
<td>• yes: label text in italics.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no: normal text.</td>
</tr>
<tr>
<td>increment</td>
<td>Optional</td>
<td>HTML</td>
<td>The unit increment value for a snapping slider.</td>
</tr>
<tr>
<td>label</td>
<td>Optional</td>
<td></td>
<td>Label to display with control; for example, &quot;Volume&quot; This displays:&quot;Volume %value%&quot;}To reference the value, use {{&quot;%value%&quot;. If %% is omitted, slider value displays directly after label.</td>
</tr>
<tr>
<td>lookAndFeel</td>
<td>Optional</td>
<td>Windows</td>
<td>• motif: renders slider using Motif style.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• windows: renders slider using Windows style.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• metal: renders slider using Java Swing style.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If platform does not support choice, the tag defaults to the platform's default style.</td>
</tr>
<tr>
<td>max</td>
<td>Optional</td>
<td>HTML</td>
<td>Maximum value for the slider.</td>
</tr>
<tr>
<td>min</td>
<td>Optional</td>
<td>HTML</td>
<td>Minimum value for the slider.</td>
</tr>
<tr>
<td>message</td>
<td>Optional</td>
<td>Applet</td>
<td>Message text to appear if validation fails.</td>
</tr>
<tr>
<td><strong>notSupported</strong></td>
<td>Optional</td>
<td>Text to display if a page that contains a Java applet-based <code>cf.form</code> control is opened by a browser that does not support Java or has Java support disabled. For example:{{&quot;&lt;b&gt;Browser must support Java to view ColdFusion Java Applets&lt;/b&gt;&quot;}} Default message:&lt;b&gt;Browser must support Java to &lt;br&gt;view ColdFusion Java Applets!&lt;/b&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>onChange</strong></td>
<td>OptionalHTML</td>
<td>Custom JavaScript function to run when slider value changes. Specify only the function name.</td>
<td></td>
</tr>
<tr>
<td><strong>onDrag</strong></td>
<td>OptionalHTML</td>
<td>Custom JavaScript function to run when you drag the slider. Specify only the function name.</td>
<td></td>
</tr>
<tr>
<td><strong>onError</strong></td>
<td>Optional</td>
<td>Custom JavaScript function to run if validation fails. Specify only the function name.</td>
<td></td>
</tr>
<tr>
<td><strong>onValidate</strong></td>
<td>Optional</td>
<td>Custom JavaScript function to validate user input; in this case, a change to the default slider value. Specify only the function name.</td>
<td></td>
</tr>
<tr>
<td><strong>range</strong></td>
<td>Optional</td>
<td>&quot;0,100&quot; Numeric slider range values. Separate values with a comma.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>scale</td>
<td>Optional</td>
<td>Unsigned integer. Defines slider scale, within range. For example, if <code>range = &quot;0,1000&quot;</code> and <code>scale = &quot;100&quot;</code>, the display values are: 0, 100, 200, 300, ... Signed and unsigned integers in ColdFusion are in the range -2,147,483,648 to 2,147,483,647.</td>
<td></td>
</tr>
<tr>
<td>textColor</td>
<td>Optional</td>
<td>Options: same as for <code>bgc</code> attribute.</td>
<td></td>
</tr>
<tr>
<td>tip</td>
<td>OptionalHTML</td>
<td>true</td>
<td>Whether the data value has to display as data tips:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• false</td>
</tr>
<tr>
<td>value</td>
<td>Optional</td>
<td>Minimum in range</td>
<td>Starting slider setting. Must be within the <code>range</code> values.</td>
</tr>
<tr>
<td>vertical</td>
<td>Optional</td>
<td>no (for applet forms) false (for HTML forms)</td>
<td>For Applet forms:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• yes: renders slider in browser vertically. Set <code>w</code> and <code>height</code> attributes; ColdFusion does not automatically swap width and height values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• no: renders slider horizontally. For HTML forms:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• true: renders slider in browser vertically. Set <code>w</code> and <code>height</code> attributes; ColdFusion does not automatically swap width and height values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• false: renders slider horizontally.</td>
</tr>
<tr>
<td>vSpace</td>
<td>Optional</td>
<td></td>
<td>Vertical spacing above and below slider, in pixels.</td>
</tr>
<tr>
<td>width</td>
<td>Optional</td>
<td>200, for HTML</td>
<td>Slider control width, in pixels.</td>
</tr>
</tbody>
</table>
Usage

This tag requires the client to download a Java applet. Using this tag is sometimes slightly slower than using an HTML form element to display the same information. Also, if the client does not have an up-to-date Java plug-in installed, the system sometimes has to download an updated Java plug-in to display the tag.

For this tag to work properly, the browser must be JavaScript-enabled.

If the following conditions are true, a user’s selection from query data that populates this tag’s options continues to display after the user submits the form:

- The `cfform preserveData` attribute is set to "Yes".
- The `cfform action` attribute posts to the same page as the form itself (this is the default), or the action page has a form that contains controls with the same names as corresponding controls on the user entry form.

For more information, see the `cfform` tag entry.

Example

```html
<!--- This example shows how to use cfslider</h3>
<br/>
<cfform name="form01">
  <cfslider name="slider1" format="HTML" vertical="false" width="350" value="100" min="0" max="200" increment="10" tip="true"/>
</cfform>
```
cfspreadsheet

Description

Manages Excel spreadsheet files:

- Reads a sheet from a spreadsheet file and stores it in a ColdFusion spreadsheet object, query, CSV string, or HTML string.
- Writes single sheet to a new XLS file from a query, ColdFusion spreadsheet object, or CSV string variable.
- Add a sheet an existing XLS file.

Category

Extensibility tags

Syntax

The tag syntax depends on the action attribute value:

```
Read
<cfspreadsheet
    action="read"
    src = "filepath"
    columns = "range"
    columnnames = "comma-delimited list"
    excludeHeaderRow = "true | false"
    format = "CSV|HTML"
    headerrow = "row number"
    name = "text"
    query = "query name"
    rows = "range"
    sheet = "number"
    sheetname = "text"/>

Update
<cfspreadsheet
    action="update"
    filename = "filepath"
    format = "csv"
    name = "text"
    password = "password"
    query = "query name"
    sheetname = "text" >

Write
<cfspreadsheet
    action="write"
    filename = "filepath"
    format = "csv"
    name = "text"
    overwrite = "true | false"
    password = "password"
    query = "queryname"
    sheetname = "text"
        autosize = "true | false" >
```
See also

Spreadsheet functions.

History

ColdFusion 11: Added the autosize attribute.

ColdFusion 9.0.1: Added the attribute {{excludeHeaderRow}}

ColdFusion 9: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>All</td>
<td>Required</td>
<td></td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• <strong>read</strong>—Reads the contents of an XLS format file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• <strong>update</strong>—Adds a new sheet to an existing XLS file. You cannot use the <strong>update</strong> action to change an existing sheet in a file. For more information, see Usage.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• <strong>write</strong>—Writes a new XLS format file or overwrites an existing file.</td>
</tr>
<tr>
<td>filename</td>
<td>{{update, write}}</td>
<td>Required</td>
<td></td>
<td>The pathname of the file that is written.</td>
</tr>
<tr>
<td>excludeHeaderRow</td>
<td>read</td>
<td>Optional</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
</tbody>
</table>

If set to `true`, excludes the headerrow from being included in the query results. The attribute helps when you read Excel as a query. When you specify the `headerrow` attribute, the column names are retrieved from the header row. But they are also included in the first row of the query. To not include the header row, set `true` as the attribute value.

<table>
<thead>
<tr>
<th>name</th>
<th>All</th>
<th>name or query is required.</th>
</tr>
</thead>
</table>

- **read action**: The variable in which to store the spreadsheet file data. Specify `name` or `query`.
- **write and update actions**: A variable containing CSV-format data or an ColdFusion spreadsheet object containing the data to write. Specify the `name` or `query`. 
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>All</td>
<td>name</td>
<td>query is required.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>query</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>required.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>src</td>
<td>read</td>
<td>Required</td>
<td>The pathname of the file to read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>columns</td>
<td>read</td>
<td>Optional</td>
<td>Column number or range of columns. Specify a single number, a hyphen-separated column range, a comma-separated list, or any combination of these; for example: 1,3-6,9.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>columnnames</td>
<td>read</td>
<td>Optional</td>
<td>Comma-separated column names.</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td><strong>value</strong></td>
<td><strong>description</strong></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>All</td>
<td>Optional</td>
<td>For read, save as a spreadsheet object. For update and write: Save a spreadsheet object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>headerrow</td>
<td>read</td>
<td>Optional</td>
<td>Row number that contains column names.</td>
</tr>
<tr>
<td>overwrite</td>
<td>write</td>
<td>Optional</td>
<td>false</td>
</tr>
<tr>
<td>password</td>
<td>updatewrite</td>
<td>Optional</td>
<td>Set a password for modifying the sheet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> Setting a password of the empty string does no unset password protection entirely; you are still prompted for a password if you try to modify the sheet.</td>
</tr>
<tr>
<td>rows</td>
<td>read</td>
<td>Optional</td>
<td>The range of rows to read. Specify a single number, a hyphen-separated row range, a comma-separated list, or any combination of these; for example: 1,3-6,9.</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>sheet</td>
<td>read</td>
<td>Optional</td>
<td>Number of the sheet. For the read action, you can specify sheet or sheetname.</td>
</tr>
<tr>
<td>sheetname</td>
<td>All</td>
<td>Optional</td>
<td>Name of the sheet. For the read action, you can specify sheet or sheetname. For write and update actions, the specified sheet is renamed according to the value you specify for sheetname.</td>
</tr>
<tr>
<td>autosize</td>
<td>write</td>
<td>Optional</td>
<td>Auto resize the column, if required. True or False.</td>
</tr>
</tbody>
</table>

**Usage**

Each ColdFusion spreadsheet object represents Excel sheet:

- To read an Excel file with multiple sheets, use multiple `cfspreadsheet` tags with the `read` option and specify different `name` and `sheet` or `sheetname` attributes for each sheet.
- To write multiple sheets to a single file, use the `write` action to create the file and save the first sheet and use the `update` action to add each additional sheet.
- To update an existing file, read all sheets in the file, modify one or more sheets, and use the contents, and use the `write` action and `Update` actions (for multiple sheet files) to rewrite the entire file.

The `cfspreadsheet` tag writes only XLS format files. To write a CSV file, put your data in a CSV formatted string variable and use the `cffile` tag to write the variable contents in a file. Use the ColdFusion Spreadsheet functions, such as `SpreadsheetNew` and `SpreadsheetAddColumn` to create a new ColdFusion Spreadsheet object and modify the spreadsheet contents.

**Example**
The following example uses the cfspreadsheet tag to read and write Excel spreadsheets using various formats. It also shows a simple use of ColdFusion Spreadsheet functions to modify a sheet.

```cfml
<!--- Read data from two datasource tables. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, COURSE_ID, CORNAME
    FROM COURSELIST
</cfquery>

<cfquery
    name="centers" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT *
    FROM CENTERS
</cfquery>

<cfscript>
    // Use an absolute path for the files. --->
    theDir=GetDirectoryFromPath(GetCurrentTemplatePath());
    theFile=theDir & "courses.xls";
    // Create two empty ColdFusion spreadsheet objects. --->
    theSheet = SpreadsheetNew("CourseData");
    theSecondSheet = SpreadsheetNew("CentersData");
    // Populate each object with a query. --->
    SpreadsheetAddRows(theSheet,courses);
    SpreadsheetAddRows(theSecondSheet,centers);
</cfscript>

<!--- Write the two sheets to a single file --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheetname="courses" overwrite=true>
    <cfspreadsheet action="update" filename="#theFile#" name="theSecondSheet"
        sheetname="centers">
SpreadsheetAddColumn(spreadsheetData, "Basic,Intermediate,Advanced,Basic,Intermediate,Advanced,Basic,Intermediate,Advanced", 3, 2, true);
</cfscript>

<!--- Write the updated Courses sheet to a new XLS file --->
<cfspreadsheet action="write" filename="#theDir#updatedFile.xls" name="spreadsheetData"
    sheetname="courses" overwrite=true>
</cfspreadsheet>

<!--- Write an XLS file containing the data in the CSV variable. --->
<!---
Using the autosize param.
--->

<cfset colList = "col1, col2, col3, col4, col5, col6, col7, col8, col9, col10">

<cfset rowCount = 100>

<cfset qryObj = QueryNew("#colList#")>
<cfset QueryAddRow(qryObj, #rowCount#)>
<cfloop from="1" to="#rowCount#" index="r">
  <cfloop from="1" to="#ListLen(colList)#" index="c">
    <cfset QuerySetCell(qryObj, #ListGetAt(colList,c)#, "some random text
r#r#c#c#", r)>
  </cfloop>
</cfloop>
</cfloop>

<cfset xlObj = spreadsheetNew("testsheet", true)>
<cfset spreadsheetAddRows(xlObj, #qryObj#)>
<cfspreadsheet action = "write" filename = 
"#Expandpath("./")#test_autosize_ss_tag.xlsx" name = "xlObj" overwrite = "true"
sheetname = "testsheet" autosize = true>
cfsprydataset

Description

Creates a Spry XML or JSON data set from the results of a bind expression.

Category

Internet protocol tags

Syntax

```
<cfsprydataset
    bind = "bind expression"
    name = "data set name"
    onBindError = "JavaScript function name"
    options = "Spry options object"
    type = "xml|json"
    xpath = "XPath expression">
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfajaximport, Using Spry with ColdFusion in the Developing ColdFusion Applications

History

ColdFusion 8: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

| **bind** | **Required** | A bind expression that returns an XML- or JSON-formatted string to populate the Spry data set. The bind expression specifies a CFC function or URL and includes bind parameters that represent the values of ColdFusion controls. For detailed information on bind expressions, see [Binding data to form fields](#) in the *Developing ColdFusion Applications*.

| **name** | **Required** | The name of the Spry data set.

| **onBindError** | **Optional; HTML** | See Description | The name of a JavaScript function to execute if the bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and specified a global error handler (by using the `ColdFusion.setGlobalErrorHandler` function), the handler displays the error message; otherwise a default error pop-up appears.

| **options** | **Optional** | A JavaScript object containing constructor options for the data set. For example, to request the data using the HTTP POST method, specify the following attribute: `options = '{method: 'POST'}'`. For detailed information on Spry options, see the Spry documentation at [www.adobe.com/go/learn_spry_framework_en](http://www.adobe.com/go/learn_spry_framework_en). |
| type | Optional | xml | Specifies data set type, corresponding to the format of the data that is returned by the bind expression. The following values are valid:
|      |          |    | - json
|      |          |    | - xml

| xpath | Required for xml type | Not used for JSON | An XPath expression that extracts data from the XML returned by the bind expression. The data set contains only the data that matches the XPath expression.

Usage

Use this tag to use a bind expression to dynamically create the contents of a Spry XML or JSON data set based on the value of a ColdFusion control or another Spry data set. To create a Spry data set without using a bind expression, use the Spry.Data.JSONDataSet() and Spry.Data.XMLDataSet() JavaScript functions. This tag cannot create a Spry HTML data set.

To use a filter to select the contents of a JSON data set from a JSON expression, specify a path or subpath option in the options attribute. For example, to create a Spry JSON data set by using only the items.item element from the JSON data, use a tag such as the following:

```xml
<cf sprayedata type="json"
    bind="CFC:GridDataManager.getProductDetails(prodname={bookform:bookgrid.TITLE})"
    options="{path: 'items.item.'})">
```

Example

The following cf sprayedataset tag updates the dsProduct Spry XML data set by calling the GridDataManager.getProductDetails CFC function each time the selected row in the bookgrid control changes. It passes the TITLE field of the selected row to the CFC function as a pro name parameter. For a complete example that uses this tag, see Using Spry with ColdFusion in the Developing ColdFusion Applications.

```xml
<cf sprayedataset
    name="dsProduct"
    type="xml"
    bind="CFC:GridDataManager.getProductDetails(prodname=
        {bookform:bookgrid.TITLE})"
    xpath="products/product"
    options="{method: 'POST'}"
    onBindError="errorHandler">
```
## cfstoredproc

### Description
Executes a stored procedure in a server database. It specifies database connection information and identifies the stored procedure.

### Category
**Database manipulation tags**

### Syntax
```
<cfstoredproc
   dataSource = "data source name"
   procedure = "procedure name"
   cachedAfter = "date"
   cachedWithin = "time span"
   debug = "yes|no"
   timeout = "timeout interval"
   fetchClientInfo = "yes|no"
   blockFactor = "block size"
   password = "password"
   result = "result name"
   returnCode = "yes|no"
   username = "user name">
```

### Note
You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

### See also
- cfinsert, cfqueryparam, cfproccparam, cfprocresult, cftransaction, cfquery, cfupdate: Optimizing database use in the Developing ColdFusion Applications

### History
- ColdFusion 11: Added the attribute, `dbVarName`.
- ColdFusion 10: Added the following attributes: `timeout`, `fetchClientInfo`, and `clientInfo`.
- ColdFusion MX 7: Added the `result` attribute.
- ColdFusion MX: Deprecated the `connectString`, `dbName`, `dbServer`, `dbtype`, `provider`, and `providerDSN` attributes. They do not work, and might cause an error, in releases later than ColdFusion 5. (Releases starting with ColdFusion MX use Type 4 JDBC drivers.)

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>dataSource</td>
<td>Required</td>
<td></td>
<td>Name of data source that points to database that contains stored procedure.</td>
</tr>
<tr>
<td>procedure</td>
<td>Required</td>
<td></td>
<td>Name of stored procedure on database server.</td>
</tr>
<tr>
<td>blockFactor</td>
<td>Optional</td>
<td>1</td>
<td>Maximum number of rows to get at a time from server. Range is 1 to 100.</td>
</tr>
<tr>
<td>cachedAfter</td>
<td>Optional</td>
<td></td>
<td>A date value (for example, April 16, 2008, 4-16-2008). If the date of original query is after this date, ColdFusion uses cached query data. To use cached data, the current query must use same SQL statement, data source, query name, user name, and password. A date/time object is in the range 100 AD-9999 AD. When specifying a date value as a string, enclose it in quotation marks.</td>
</tr>
<tr>
<td>cachedWithin</td>
<td>Optional</td>
<td></td>
<td>A time span, created using the <code>CreateTimeSpan</code> function. If the original query date falls within the time span, cached query data is used. <code>CreateTimeSpan</code> defines a period from the present, back. Takes effect only if query caching is enabled in the Administrator. To use cached data, the current query must use the same SQL statement, data source, query name, user name, and password.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>-------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>clientInfo</td>
<td>Optional</td>
<td>Structure containing properties of the client to be set on the database connection.</td>
<td></td>
</tr>
<tr>
<td>debug</td>
<td>Optional</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: lists debug information on each statement.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no</td>
<td></td>
</tr>
<tr>
<td>fetchClientInfo</td>
<td>Optional</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If set to yes, returns a struct with the key-value pair passed by the last query.</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides password in data source setup.</td>
<td></td>
</tr>
<tr>
<td>result</td>
<td>Optional</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifies a name for the structure in which cfstroredproc returns the statusCode and ExecutionTime variables. If set, this value replaces cfstoredproc as the prefix to use when accessing those variables. For more information, see Usage.</td>
<td></td>
</tr>
<tr>
<td>returnCode</td>
<td>Optional</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: populates cfstoredproc.statusCode with status code returned by the stored procedure.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no</td>
<td></td>
</tr>
<tr>
<td>timeOut</td>
<td>Optional</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of seconds each action is permitted to execute, before returning an error. The cumulative time may exceed this value. For JDBC statements, ColdFusion sets this attribute. For other drivers, see the driver documentation.</td>
<td></td>
</tr>
<tr>
<td>username</td>
<td>Optional</td>
<td>Overrides username in data source setup.</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>dbVarName</td>
<td>Optional</td>
<td>When sending the named parameters to a stored procedure, this attribute can be used to specify the name of the parameter.</td>
<td></td>
</tr>
</tbody>
</table>

To enable dbVarName for a datasource, add the system property `coldfusion.dbvarDependants`.

For example, if the list of DSNs that require the dbVarName support are dsn1 and dsn2, then add the JVM flag:

```
-Dcoldfusion.dbvarDependants={dsn1, dsn2}
```

Usage

Use this tag to call a database stored procedure. Within this tag, you code `cfprocresult` and `cfprocparam` tags as follows:

- **cfprocresult**: If the stored procedure returns one or more result sets, code one `cfprocresult` tag per result set.
- **cfprocparam**: If the stored procedure uses input or output parameters, code one `cfprocparam` tag per parameter, ensuring that you include every parameter in the stored procedure definition.

If you set `returnCode = "Yes"`, this tag sets the variable `prefix.statusCode`, which holds the status code for a stored procedure. Status code values vary by DBMS. For the meaning of code values, see your DBMS documentation. This tag sets the variable `prefix.ExecutionTime`, which contains the execution time of the stored procedure, in milliseconds.

The value of `prefix` is either `cfstoredproc` or the value specified by the `result` attribute, if it is set. The `result` attribute provides a way for stored procedures that are called from multiple pages, possibly at the same time, to avoid overwriting the results of one call with another. If you set the `result` attribute to `myResult`, for
example, you would access `ExecutionTime as myResult.ExecutionTime`. Otherwise, you would access it as `cfstoredproc.ExecutionTime`.) Before implementing this tag, ensure that you understand stored procedures and their usage. The following examples use a Sybase stored procedure; for an example of an Oracle 8 or 9 stored procedure, see `cfprocparam`.

Example

```cfml
<cfset ds = "sqltst">

<!---
If submitting a new book, insert the record and display confirmation --->
<cfif isDefined("form.title")>

<cfstoredproc procedure="Insert_Book" datasource="#ds#">

<cfprocparam
cfsqltype="cf_sql_varchar"
value="#form.title#">

<cfprocparam
cfsqltype="cf_sql_numeric"
value="#form.price#">

<cfprocparam
cfsqltype="cf_sql_date"
value="#form.publishDate#">

<cfprocparam
cfsqltype="cf_sql_numeric" type="out" variable="bookId">

</cfstoredproc>

<cfoutput>
<h3>'#form.title#' inserted into database. The ID is #bookId#.</h3>
</cfoutput>

</cfif>

</cfif>

<cfform action="#CGI.SCRIPT_NAME#" method="post">
<h3>Insert a new book</h3>

Title:
<cfinput type="text" size="20" required="yes" name="title"/>
<br/>

Price:
<cfinput type="text" size="20" required="yes" name="price" validate="float" />
<br/>

Publish Date:
<cfinput type="text" size="5" required="yes" name="publishDate" validate="date" />
<br/>
```
<!---
This view-only example executes a Sybase stored procedure that returns three result sets, two of which we want. The stored procedure returns the status code and one output parameter, which we display. We use named notation for the parameters.
--->
<cfstoredproc procedure = "foo_proc"
    dataSource = "MY_SYBASE_TEST" username = "sa"
    password = "" dbServer = "scup" dbName = "pubs2"
    returnCode = "Yes" debug = "Yes">
    <cfprocresult name = RS1>
    <cfprocresult name = RS3 resultSet = 3>
    <cfprocparam type = "IN" CFSQLType = CF_SQL_INTEGER
        value = "1" dbVarName = @param1>
    <cfprocparam type = "OUT" CFSQLType = CF_SQL_DATE
        variable = FOO dbVarName = @param2>
</cfstoredproc>

--->
<cfoutput> The output param value: #foo#</cfoutput>
<h3>The Results Information</h3>
<cfoutput query = RS1>#name#, #DATE_COL#</cfoutput>
<p>
<cfoutput>
<hr>
<p>Record Count: #RS1.recordCount# Columns: #RS1.columnList#
</cfoutput>
<cfoutput query = RS3>#col1#, #col2#, #col3#</cfoutput>
<p>
<cfoutput>
<hr>
<p>Record Count: #RS3.recordCount# Columns: #RS3.columnList#
</cfoutput>
The return code for the stored procedure is: #cfstoredproc.statusCode#
</cfoutput>

---


**cfswitch**

**Description**

Evaluates a passed expression and passes control to the `cfcase` tag that matches the expression result. You can, optionally, code a `cfdefaultcase` tag, which receives control if there is no matching `cfcase` tag value.

**Category**

*Flow-control tags*

**Syntax**

```cftags
<cfswitch
   expression = "expression">
   one or more cfcase tags
   zero or one cfdefaultcase tags
</cfswitch>
```

**See also**

`cfcase`, `cfdefaultcase`, `cfabort`, `cfloop`, `cfbreak`, `cfexecute`, `cfexit`, `cfif`, `cflocation`, `cfrethrow`, `cfthrow`, `cftry`; `cfswitch`, `cfcase`, and `cfdefaultcase` in the *Developing ColdFusion Applications*

**History**

ColdFusion 8: Changed the way the ColdFusion parses `<cfcase>` values. Previously, `<cfcase>` tags with numeric value dates did not return expected results. For example, `<cfcase value="00">` and `<cfcase value="0A">` were both evaluated to 0. The value "0A" was treated as a date and converted to 0 number of days from 12/30/1899. The value "00" was also evaluated to the value 0. This caused the exception "Context validation error for tag CFCASE. The CF SWITCH has a duplicate CF CASE for value "0.0"." The `<cfswitch>` tag now returns the expected result.

ColdFusion MX: Changed `cfdefaultcase` tag placement requirements: you can put the `cfdefaultcase` tag at any position within a `cfswitch` statement; it is not required to be the last item.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>Required</td>
<td></td>
<td>ColdFusion expression that yields a scalar value. ColdFusion converts integers, real numbers, Booleans, and dates to numeric values. For example, <code>true</code>, 1, and 1.0 are all equal.</td>
</tr>
</tbody>
</table>

**Usage**

This tag requires an end tag. All code within this tag must be within a `cfcase` or `cfdefaultcase` tag. Otherwise,
ColdFusion throws an error. Use this tag followed by one or more `cfcase` tags. Optionally, include a `cfdefaultcase` tag. This tag selects the matching alternative from the `cfcase` and `cfdefaultcase` tags, jumps to the matching tag, and executes the code between the `cfcase` start and end tags. The `cfswitch` tag provides better performance than a series of `cfif/cfelseif` tags, and the code is easier to read.

**Example**

```cfml
<!---
This example shows the use of `cfswitch` and `cfcase` to
execute a case statement in CFML.
--->
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
  SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
  FROM Employees
</cfquery>

<h3>cfswitch Example</h3>
<!--- By outputting the query and using `cfswitch`, we classify the
output without using a `cfloop` construct. --->
<p>Each time the case is fulfilled, the specific information is printed;
if the case is not fulfilled, the default case is output</p>
<cfoutput query="GetEmployees">
  <cfswitch expression="#Trim(Department)#">
    <cfcase value="Sales">
      #FirstName# #LastName# is in <b>sales</b><br>
    </cfcase>
    <cfcase value="Accounting">
      #FirstName# #LastName# is in <b>accounting</b><br>
    </cfcase>
    <cfcase value="Administration">
      #FirstName# #LastName# is in <b>administration</b><br>
    </cfcase>
    <cfdefaultcase>
      #FirstName# #LastName# is not in Sales, Accounting, or Administration.<br>
    </cfdefaultcase>
  </cfswitch>
</cfoutput>
```

#back to top
This section lists tags starting with alphabet T with their description.

- cftable
- cftextarea
- cftextinput
- cfthread
- cfthrow
- cftimer
- cftooltip
- cftrace
- cftransaction
- cftree
- cfitem
- cftry
cftable

Description

Builds a table in a ColdFusion page. This tag renders data as preformatted text, or, with the HTMLTable attribute, in an HTML table. If you don't want to write HTML table tag code, or if your data can be presented as preformatted text, use this tag. Preformatted text (defined in HTML with the <pre> and </pre> tags) displays text in a fixed-width font. It displays white space and line breaks exactly as they are written within the pre tags. For more information, see an HTML reference guide.
To define table column and row characteristics, use the cfcol tag within this tag.

Category

Data output tags

Syntax

```cftable
    query = "query name"
    border
    colHeaders
    colSpacing = "number of spaces"
    headerLines = "number of lines"
    HTMLTable
    maxRows = "maxrows table"
    startRow = "row number">
    ...
</cftable>
```

⚠️ Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfcol, cfcontent, cflog, cfoutput, cfprocessingdirective; Retrieving data in the Developing ColdFusion Applications

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>Required</td>
<td></td>
<td>Name of cfquery from which to draw data.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>border</td>
<td>Optional</td>
<td></td>
<td>Displays a border around the table. If you use this attribute (regardless of its value), ColdFusion displays a border around the table. Use this only if you use the <code>HTMLTable</code> attribute.</td>
</tr>
<tr>
<td>colHeaders</td>
<td>Optional</td>
<td></td>
<td>Displays column heads. If you use this attribute, also use the <code>cfcol</code> tag <code>header</code> attribute to define them. If you use this attribute (regardless of its value), ColdFusion displays column heads.</td>
</tr>
<tr>
<td>colSpacing</td>
<td>Optional</td>
<td>2</td>
<td>Number of spaces between columns.</td>
</tr>
<tr>
<td>headerLines</td>
<td>Optional</td>
<td>2</td>
<td>Number of lines to use for table header (the default leaves one line between header and first row of table).</td>
</tr>
<tr>
<td>HTMLTable</td>
<td>Optional</td>
<td></td>
<td>Renders data in an HTML 3.0 table. If you use this attribute (regardless of its value), ColdFusion renders data in an HTML table.</td>
</tr>
<tr>
<td>maxRows</td>
<td>Optional</td>
<td></td>
<td>Maximum number of rows to display in the table.</td>
</tr>
<tr>
<td>startRow</td>
<td>Optional</td>
<td>1</td>
<td>The query result row to put in the first table row.</td>
</tr>
</tbody>
</table>

**Usage**

This tag aligns table data, sets column widths, and defines column heads. At least one `cfcol` tag is required within this tag. Put `cfcol` and `cftable` tags adjacent in a page. The only tag that you can nest within this tag is the `cfcol` tag. You cannot nest `cftable` tags.

To display the `cfcol` header text, specify the `cfcol header` and the `cftable colHeader` attribute. If you specify either attribute without the other, the header does not display and no error is thrown.

**Example**
<!-- This example shows the use of cfcol and cftable to align information returned from a query. -->

<!-- This query selects employee information from cfdocexamples datasource. -->
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
  SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
  FROM Employees
</cfquery>

<html>
<body>
<h3>cftable Example</h3>

<!-- Note use of HTMLTable attribute to display cftable as an HTML table, rather than as PRE formatted information. -->
<cftable query = "GetEmployees"
  startRow = "1" colSpacing = "3" HTMLTable>
  <!-- Each cfcol tag sets width of a column in table, and specifies header information and text/CFML with which to fill cell. -->
  <cfcol header = "<b>ID</b>"
    align = "Left"
    width = 2
    text = "#Emp_ID#">
  <cfcol header = "<b>Name/Email</b>"
    align = "Left"
    width = 15
    text = "<a href = 'mailto:#Email#'>#FirstName# #LastName#</a>">
  <cfcol header = "<b>Phone Number</b>"
    align = "Center"
    width = 15
    text = "#Phone#">
</cftable>
</body>
</html>
cftextarea

Description

Puts a multiline text entry box in a cfform tag and controls its display characteristics. Optionally, displays a rich text editor with configurable controls for formatting HTML text.

Category

Forms tags

Syntax
<cftextarea
  name="name"
  basePath="path"
  bind="bind expression"
  bindAttribute="attribute name"
  bindOnLoad="false|true"
  disabled="true|false" or no attribute value
  enabled="yes|no"
  fontFormats="comma separated list"
  fontNames="comma separated list"
  fontSizes="comma separated list"
  height="number of pixels"
  html="no|yes"
  label="text"
  maxLength="number"
  message="text"
  onBindError = "JavaScript function name"
  onChange="JavaScript or ActionScript"
  onClick="JavaScript or ActionScript"
  onError="script name"
  onKeyPress="JavaScript or ActionScript"
  onMouseDown="JavaScript or ActionScript"
  onMouseUp="JavaScript or ActionScript"
  onValidate="script name"
  pattern="regexp"
  range="minimum value, maximum value"
  required="yes|no"
  richtext="no|yes"
  secureUpload="true|false"
  skin="default|silver|office2003|custom skin"
  sourceForToolTip="URL"
  style="style specification"
  stylesXML="path"
  templatesXML="path"
  toolbar="Default|Basic|custom toolbar"
  toolbarOnFocus="no|yes"
  tooltip="tip text"
  validate="data type"
  validateAt= one or more of "onBlur, onServer, onSubmit"
  value="text"
  visible="yes|no"
  width="number of pixels"
  wrap="off|hard|soft|virtual|physical">
  text
</cftextarea>

Some attributes apply to only specific display formats. For details see the Attributes table.
Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also
cfajaximport, cfapplet, cfcalendar, cfform, cfformgroup, cfformitem, cfgrid, cfinput, cfselect, cfslider, cftree; Introduction to Retrieving and Formatting Data and Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8:

- Added support for the bind attribute in HTML format forms, including the bindAttribute, bindOnLoad, and onBindError attributes.
- Added support for tool tips in HTML format, including the sourceForTooltip attribute.
- Added support for a rich text editor in HTML format, including the richtext, basepath, fontFormats, fontNames, fontSizes, skin, stylesXML, templatesXML, toolbar, and toolbarOnFocus attributes and support for the height and width attributes.

ColdFusion MX 7: Added this tag.

Attributes

The following table lists attributes that ColdFusion uses directly. In HTML format, the tag also supports all HTML textarea tag attributes that are not on this list, and passes them directly to the browser.

Note

Attributes that are not marked as All or XML are not handled by the skins provided with ColdFusion. They are, however, included in the generated XML.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt;Format</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required;All</td>
<td></td>
<td>Name of the cftextinput control.</td>
</tr>
<tr>
<td>basepath</td>
<td>Optional; HTML</td>
<td>/CFIDE/scripts/ajax/FCKEditor</td>
<td>Path to the directory that contains the rich text editor. The editor configuration files are at the top level of this directory. Meaningful only if the richText attribute is true.</td>
</tr>
<tr>
<td><strong>bind</strong></td>
<td>Optional; Flash, HTML</td>
<td>A bind expression that dynamically sets an attribute of the control. For details, see Usage.</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>bindAttribute</strong></td>
<td>Optional; HTML</td>
<td>value</td>
<td>Specifies the HTML tag attribute whose value is set by the <code>bind</code> attribute. You can only specify attributes in the browser's HTML DOM tree, not ColdFusion-specific attributes. Ignored if there is no <code>bind</code> attribute.</td>
</tr>
<tr>
<td><strong>bindOnLoad</strong></td>
<td>Optional; HTML</td>
<td>no</td>
<td>A Boolean value that specifies whether to execute the <code>bind</code> attribute expression when first loading the form. Ignored if there is no <code>bind</code> attribute.</td>
</tr>
<tr>
<td><strong>disabled</strong></td>
<td>Optional; All</td>
<td>not disabled</td>
<td>Disables user input, making the control read-only. To disable input, specify <code>disabled</code> without an attribute, or <code>disabled=&quot;Yes&quot;</code> (or any ColdFusion positive Boolean value, such as True). To enable input, omit the attribute or specify <code>disabled=&quot;No&quot;</code> (or any ColdFusion negative Boolean value, such as False).</td>
</tr>
<tr>
<td><strong>enabled</strong></td>
<td>Optional; Flash</td>
<td>yes</td>
<td>Boolean value that specifies whether the control is enabled. A disabled control appears in light gray. The inverse of the <code>disabled</code> attribute.</td>
</tr>
<tr>
<td><strong>fontFormats</strong></td>
<td>Optional; HTML</td>
<td>All valid formats</td>
<td>A comma-separated list of the font names to display in the rich text editor Formats selector. The formats specify the HTML tags to apply to typed or selected text. You can specify any subset of the following list: &quot;p,div,pre,address,h1,h2,h3,h4,h5,h6&quot;.</td>
</tr>
<tr>
<td><strong>fontNames</strong></td>
<td>Optional; HTML</td>
<td>All valid font names</td>
<td>A comma-separated list of the font names to display in the rich text editor Font selector. You can specify any subset of the following list: &quot;Arial,Comic Sans MS,Courier New,Tahoma,Times New Roman,Verdana&quot;.</td>
</tr>
<tr>
<td><strong>FontSizes</strong></td>
<td>Optional; HTML</td>
<td>See Description</td>
<td>A comma-separated list of the font sizes to display in the rich text editor Size selector. List entries must have the format of <em>numeric font size</em>/descriptive text. For example, to display the text &quot;small font&quot; to indicate a font size of 1, specify &quot;1/small font&quot;. By default, the following values appear in the selector: 1/xx-small,2/x-small,3/small,4/medium,5/large,6/x-large,7/xx-large.</td>
</tr>
<tr>
<td><strong>height</strong></td>
<td>Optional; Flash,HTML</td>
<td></td>
<td>In Flash forms, height of the control, in pixels. In HTML forms, this attribute has an effect only if you specify <code>richtext=&quot;true&quot;</code>; in this case, it is the height, in pixels, of the control, including the control bar and text box.</td>
</tr>
<tr>
<td>attribute</td>
<td>type</td>
<td>value</td>
<td>description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>html</td>
<td>Optional; Flash</td>
<td>no</td>
<td>Boolean value that specifies whether the text area contains HTML.</td>
</tr>
<tr>
<td>label</td>
<td>Optional; Flash and XML</td>
<td></td>
<td>Label to put beside the control on a form.</td>
</tr>
<tr>
<td>maxLength</td>
<td>Optional; All</td>
<td></td>
<td>The maximum length of text that can be entered. ColdFusion uses this attribute only if you specify <code>maxlength</code> in the <code>validate</code> attribute.</td>
</tr>
<tr>
<td>message</td>
<td>Optional; All</td>
<td></td>
<td>Message text to display if validation fails.</td>
</tr>
<tr>
<td>onBindError</td>
<td>Optional; HTML</td>
<td>See Description</td>
<td>The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. (The status code is -1 if the error is not an HTTP error.) If you omit this attribute, and have specified a global error handler (by using the <code>ColdFusion.setGlobalErrorHandler</code> function), it displays the error message; otherwise a default error pop-up displays.</td>
</tr>
<tr>
<td>onChange</td>
<td>Optional; All</td>
<td></td>
<td>JavaScript (HTML/XML) or ActionScript (Flash) to run when the control changes due to user action.</td>
</tr>
<tr>
<td>onClick</td>
<td>Optional; HTML and XML</td>
<td></td>
<td>JavaScript to run when the user clicks the control.</td>
</tr>
<tr>
<td>onError</td>
<td>Optional; HTML and XML</td>
<td></td>
<td>Custom JavaScript function to execute if validation fails.</td>
</tr>
<tr>
<td>Event</td>
<td>Required/Optional</td>
<td>Technology</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>onKeydown</td>
<td>Optional;All</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash)</td>
<td>ActionScript to run when the user presses a keyboard key in the control.</td>
</tr>
<tr>
<td>onKeyUp</td>
<td>Optional;All</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash)</td>
<td>ActionScript to run when the user releases a keyboard key in the control.</td>
</tr>
<tr>
<td>onMouseDown</td>
<td>Optional;All</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash)</td>
<td>ActionScript to run when the user releases a mouse button in the control.</td>
</tr>
<tr>
<td>onMouseUp</td>
<td>Optional;All</td>
<td>JavaScript (HTML/XML) or ActionScript (Flash)</td>
<td>ActionScript to run when the user presses a mouse button in the control.</td>
</tr>
<tr>
<td>onValidate</td>
<td>Optional;HTML and XML</td>
<td>Custom JavaScript function to validate user input. The JavaScript DOM form object, input object, and input object value are passed to the function, which must return <code>true</code> if validation succeeds, <code>false</code> otherwise. If you specify this attribute, ColdFusion ignores the <code>validate</code> attribute.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Syntax</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>pattern</td>
<td>Required if {{$validate = &quot;regular_expression&quot;}} HTML and XML.</td>
<td></td>
<td>JavaScript regular expression pattern to validate input. Omit leading and trailing slashes. ColdFusion uses this attribute only if you specify regex in the validate attribute. For examples and syntax, see <em>Building Dynamic Forms with cfform Tags</em> in the <em>Developing ColdFusion Applications</em>.</td>
</tr>
<tr>
<td>range</td>
<td>Optional; All</td>
<td></td>
<td>Minimum and maximum allowed numeric values. ColdFusion uses this attribute only if you specify range in the validate attribute. If you specify a single number or a single number followed by a comma, it is treated as a minimum, with no maximum. If you specify a comma followed by a number, the maximum is set to the specified number, with no minimum.</td>
</tr>
<tr>
<td>required</td>
<td>Optional; All</td>
<td>no</td>
<td>• yes: the field must contain text. • no: the field can be empty.</td>
</tr>
<tr>
<td>richText</td>
<td>Optional; HTML</td>
<td>no</td>
<td>A Boolean value specifying whether this control is a rich text editor with tool bars to control the text formatting. For detailed information on using the rich text editor, see <em>Using the rich text editor</em> in <em>Using Ajax form controls and features</em> in the <em>Developing ColdFusion Applications</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td><strong>skin</strong></td>
<td>Optional; HTML</td>
<td>default</td>
<td>Specifies the skin to be used for the rich text editor. By default, the valid values are default, silver, and office2003. You can also create custom skins that you can then specify in this attribute. For more information, see <em>Using the rich text editor</em> in <em>Using Ajax form controls and features</em> in the <em>Developing ColdFusion Applications</em>.</td>
</tr>
<tr>
<td><strong>sourceForTooltip</strong></td>
<td>Optional; HTML</td>
<td></td>
<td>The URL of a page to display as a tool tip. The page can include CFML and HTML to control the contents and format, and the tip can include images. If you specify this attribute, an animated icon appears with the text “Loading…” while the tip is being loaded.</td>
</tr>
<tr>
<td><strong>style</strong></td>
<td>Optional; All</td>
<td></td>
<td>In HTML or XML format forms, ColdFusion passes the style attribute to the browser or XML. In Flash format forms, must be a style specification in CSS format, with the same syntax and contents as used in Flex for the corresponding Flash element.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stylesXML</td>
<td>The path of the file that defines the styles in the rich text editor Styles selector. Relative paths start at the directory that contains the fckeditor.html file, normally <code>cf_webRoot/CFIDE/scripts/ajax/FCKeditor/editor</code>. You can specify an absolute path starting at the web root, such as <code>/myApps/RTEeditor.mystyles.xml</code>. For information on configuring styles, see Using the rich text editor in Using Ajax form controls and features in the Developing ColdFusion Applications.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>templatesXML</td>
<td>The path of the file that defines the templates that are displayed when you click the rich text editor Templates icon. For pathing details, see stylesXML. For information on configuring templates, see Using the rich text editor in Using Ajax form controls and features in the Developing ColdFusion Applications.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>toolbar</td>
<td>Specifies the rich text editor toolbar. By default, the valid values for this attribute are: <code>Default</code>, a complete set of controls, and <code>Basic</code>, a minimal configuration. You can add configurations; for details see Using the rich text editor in Using Ajax form controls and features in the Developing ColdFusion Applications. Note: This attribute's value is case sensitive.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>toolbarOnFocus</td>
<td>Optional; HTML</td>
<td>no</td>
<td>A Boolean value that specifies whether the rich text editor toolbar expands and displays its controls only when the rich text editor has the focus.</td>
</tr>
<tr>
<td>tooltip</td>
<td>Optional; Flash, HTML</td>
<td></td>
<td>Text to display when the mouse pointer hovers over the control. Can include HTML formatting. Ignored if you specify a sourceForTooltip attribute.</td>
</tr>
<tr>
<td>validate</td>
<td>Optional; All</td>
<td></td>
<td>The type or types of validation to perform. Available validation types and algorithms depend on the format. For details, see the Usage section of the cfinput tag reference.</td>
</tr>
<tr>
<td>validateAt</td>
<td>Optional; HTML and XML</td>
<td>onSubmit</td>
<td>How to do the validation; one or more of the following:</td>
</tr>
</tbody>
</table>
|               |                    |         | • onSubmit  
|               |                    |         | • onServer  
|               |                    |         | • onBlur  
<p>|               |                    |         | For Flash format forms, onSubmit and onBlur are identical; for both, validation is done when the user submits the form. For multiple values, use a comma-delimited list. For details, see the Usage section of the cfinput tag reference. |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Optional;All</td>
<td></td>
<td>Initial value to display in text control. You can specify an initial value as an attribute or in the tag body, but not in both places. If you specify the value as an attribute, put the closing <code>&lt;cftextarea&gt;</code> tag immediately after the opening <code>cftextarea</code> tag, with no spaces or line feeds between, or place a closing slash at the end of the opening <code>cftextarea</code> tag; for example <code>&lt;cftextarea name=&quot;description&quot; value=&quot;Enter a description.&quot; /&gt;</code>.</td>
</tr>
<tr>
<td>visible</td>
<td>Optional;Flash</td>
<td>yes</td>
<td>Boolean value that specifies whether to show the control. Space that would be occupied by an invisible control is blank.</td>
</tr>
<tr>
<td>width</td>
<td>Optional;Flash,HTML</td>
<td></td>
<td>The width of the control, in pixels. In HTML forms, this attribute has an effect only if you specify <code>richtext=&quot;true&quot;</code>.</td>
</tr>
</tbody>
</table>
| wrap    | Optional;All          |         | • hard: wraps long lines, and sends the carriage return to the server.  
• off: does not wrap long lines.  
• physical: wraps long lines, and transmits the text at all wrap points.  
• soft: wraps long lines, but does not send the carriage return to the server.  
• virtual: wraps long lines, but does not send the carriage return to the server. |

**Usage**
For this tag to work properly in HTML format, the browser must be JavaScript-enabled. If you put text in the tag body, the control displays all text characters between the cftextarea opening and closing tags; therefore, if you use line feeds or white space to format your source text, they appear in the control. If the cfform preserveData attribute is "yes", and the form posts back to the same page, the posted value (not the value of the value attribute) of the cftextinput control is used.

Validation

For a detailed description of the validation attribute and the types of validation supported by ColdFusion, see the Usage section of the cfinput tag reference. For more details on ColdFusion validation techniques, see Validating data in the Developing ColdFusion Applications.

Flash form data binding

The bind attribute lets you populate form fields using the contents of other form fields. To specify text from another Flash form field in a cftextarea bind attribute, use the following format:{sourceTagName.text}For example, the following line uses the value of the text that the user enters in the from the userName field in the greeting in the comment text box. The user can change or replace this message with a typed entry.

```
<cfform format="flash" height="300">
  <cfformitem type="text">
    Enter your name here</cfformitem>
  <cftextarea name="userName" height="20" Width="500"/>
  <cftextarea name="comment" height="100" Width="500" bind="Hello {userName.text}! Enter your comments here." />
</cfform>
```

HTML form data binding

The bind attribute lets you set any cftextarea attribute dynamically from the value of another form control or by calling a CFC or JavaScript function. By default it sets the control's value attribute, but you can specify a different attribute to set by using the bindAttribute attribute. For more information on binding, see Binding data to form fields in the Developing ColdFusion Applications.

Example

This example has two cftextarea controls. When you submit the form, ColdFusion copies the text from the first control into the second. The onBlur maxlength validation prevents you from entering more than 100 characters. The > character that closes the cftextarea opening tag, the text in the tag body, and the cftextarea closing tag are on a single line to ensure that only the desired text displays, but the line is split in this example for formatting purposes.
<h3>cftextarea Example</h3>
<cfparam name="text2" default=""/>
<cfif isdefined("form.textarea1") AND (form.textarea1 NEQ ")">
  <cfset text2=form.textarea1>
</cfif>

<cfform name="form1">
  <cftextarea name="textarea1" wrap="virtual" rows="5" cols="25" validate="maxlength" validateAt="onBlur" maxlength="100">
Replace this text. Maximum length is 100 Characters, and this text is currently 99 characters long.</cftextarea>
  <cftextarea name="textarea2" wrap="virtual" rows="5" cols="50" value="#text2#"/>
</cfform>
<input type="submit" value="submit field">
cftextinput

Description

Puts a single-line text entry box in a cfform tag and controls its display characteristics. This tag is deprecated, and is not supported in XML format forms. In its place, you must use a cfinput or cftextarea tag and use a cascading style sheet (CSS) to specify the text style characteristics.

History

ColdFusion MX 7: This tag is deprecated. In later releases it might not work, and might cause an error.

ColdFusion MX 6.1: Changed the validate = "creditcard" option requirements: the text entry must have 13-16 digits.
**cfthread**

**Description**

The *cfthread* tag enables you to create *threads*, independent streams of code execution, in your ColdFusion application. You use this tag to run or end a thread, temporarily stop thread execution, or join together multiple threads.

**Category**

*Application framework tags*

**Syntax**

```
join
<cfthread
  required
  name="thread name[,thread name]..."
  optional
  action="join"
  timeout="milliseconds"/>

run
<cfthread
  required
  name="thread name"
  optional
  action="run"
  priority="NORMAL|HIGH|LOW"
  zero or more application-specific attributes>
    Thread code
  </cfthread>

sleep
<cfthread
  required
  action="sleep"
  duration="milliseconds"/>

terminate
<cfthread
  required
  action="terminate"
  name="thread name"/>
```

For all actions except *run*, the *cfthread* tag must have an empty body and be followed immediately by a *<cfthread>* end tag, or must have no end tag and have a slash before the tag closure, as in *<cfthread action="sleep" duration="1000"/>*. 

© 2014 Adobe Systems Incorporated. All rights reserved.
Note

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

See also

`Sleep`, `Using ColdFusion Threads` in the *Developing ColdFusion Applications*

History

ColdFusion 8: Added this tag

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Applies to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Optional</td>
<td>run</td>
<td>All</td>
<td>The action to take, one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• <code>join</code>: Makes the current thread wait until the thread or threads specified in the <code>name</code> attribute complete processing, or until the period specified in the <code>timeout</code> attribute passes, before continuing processing. If you don't specify a timeout and the thread you are joining to doesn't finish, the current thread also cannot finish processing.</td>
</tr>
</tbody>
</table>
### cfthread Tag:

- **run**: Creates a thread and starts it processing. Code in the `cfthread` tag body runs simultaneously and independently of page-level code and code in other `cfthread` tags.
- **sleep**: Suspends the current thread's processing for the time specified by the `duration` attribute. Equivalent to the `Sleep` function.
- **terminate**: Stops processing of the thread specified in the `name` attribute. If you terminate a thread, the thread scope includes an ERROR metadata structure with information about the termination.

<p>| <strong>duration</strong> | <strong>Required</strong> | <strong>sleep</strong> | The number of milliseconds for which to suspend thread processing. |</p>
<table>
<thead>
<tr>
<th>name</th>
<th>Optional, Required, if action = &quot;join&quot; or &quot;terminate&quot;</th>
<th>join run terminate</th>
<th>The name of the thread to which the action applies:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>• terminate: The name of the thread to stop.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• join: The name of the thread or threads to join to the current thread. To specify multiple threads, use a comma-delimited list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• run: The name to use to identify the thread being created.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>priority</th>
<th>Optional</th>
<th>NORMAL</th>
<th>run</th>
<th>The priority level at which to run the thread. The following values are valid:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HIGH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• LOW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• NORMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Higher priority threads get more processing time than lower priority threads.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Page-level code, the code that is outside of cfthread tags, always has NORMAL priority.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional</td>
<td>0</td>
<td>join</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>---</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The number of milliseconds that the current thread waits for the thread or threads being joined to finish. If any thread does not finish by the specified time, the current thread proceeds. If the attribute value is 0, the following action occurs:

- The current thread continues waiting until all joining threads finish.
- If the current thread is the page thread, the page continues waiting until the threads are joined, even if you specify a page time-out.

Usage

Page-level code (code outside any `cfthread` tags) executes in its own thread, referred to as the page thread. Only the page thread can create other threads. A thread that you create cannot create a child thread.

⚠ **Note**

If a thread never completes processing (is hung), it continues to occupy system resources. You can use the ColdFusion Server Monitor to check for and terminate hung threads.

ColdFusion makes a complete (deep) copy of all the attribute variables before passing them to the thread, so the values of the variables inside the thread are independent of the values of any corresponding variables in other threads, including the page thread. Thus, the values passed to threads are thread safe because the attribute values cannot be changed by any other thread.

Thread scopes

Each thread has three special scopes:

- The thread-local scope is an implicit scope that contains variables that are available only to the thread, and exist only for the life of the thread.
- The Thread scope is available to the page and to all other threads started from the page. Its data remains available until the page and all threads started from the page finish, even if the page finishes before the threads complete processing.
The Attributes scope contains attributes that are passed to the scope, and is available only within the thread and only for the life of the thread. For detailed information about using ColdFusion scopes in threads, see Using ColdFusion Threads in the Developing ColdFusion Applications. All threads in a page share a single Variables scope, so you can use it for data that is common across all threads. You must be careful to lock access to the variables, if necessary, to prevent deadlocks or race conditions between threads.

**Note**

When ColdFusion uses a connector to access the web server, after the page gets completed, the CGI and Request scopes are not accessible to threads that you create by using the `cfthread` tag. This limitation does not apply if you use the integrated web server or if you run ColdFusion as a J2EE application.

### Metadata variables

The thread scope contains the following variables that provide information about the thread (metadata):

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElapsedTime</td>
<td>The amount of processor time that was spent handling the thread.</td>
</tr>
<tr>
<td>Error</td>
<td>The structure that is generated if an error occurs during thread execution. The structure contains the keys that you can access in a <code>cfcatch</code> tag. If an error occurs in a thread, page-level processing is not affected, and ColdFusion does not generate an error message. The thread with the error terminates and the page-level code or other threads can get the error information from the Error field and handle the error appropriately. For detailed information, see Handling ColdFusion thread errors in Working with threads in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>Name</td>
<td>The thread name.</td>
</tr>
<tr>
<td>Output</td>
<td>Output that is generated by the thread. A thread cannot display output; page-level code must use this variable to display thread results. For detailed information, see Handling thread output in Working with threads in the Developing ColdFusion Applications.</td>
</tr>
</tbody>
</table>
| Priority   | The thread processing priority, as specified in the `cfthread` tag attribute. The following values are valid:  
  - **HIGH**  
  - **LOW**  
  - **NORMAL** |
### Starttime
The time at which the thread began processing, in ColdFusion date-time format.

### Status
The current status of the thread; one of the following values:
- **NOT_STARTED**: The thread has been queued but is not processing yet.
- **RUNNING**: The thread is running normally.
- **TERMINATED**: The thread stopped running due to a cfthread tag with a terminate action, an error, or an administrator action.
- **COMPLETED**: The thread ended normally.
- **WAITING**: The thread has executed a cfthread tag with action="join", but one or more threads being joined has not completed.

### Example
The following example uses three threads to get the results of three RSS feeds. The user must submit the form with all three feeds specified. The application joins the threads with a time-out of 6 seconds, and displays the feed titles and the individual item titles as links.
<cfif isDefined("Form.submit")>
  <cfloop index="i" from="1" to="3">
    <!--- Use array notation and string concatenation to create a variable for this feed. --->
    <cfset theFeed = Form["Feed"&i]>
    <cfif theFeed NEQ "">
      <!--- Use a separate thread to get each of the feeds. --->
      <cfthread action="run" name="t#i#" feed="#theFeed#">
        <cffeed source="#feed#" properties="thread.myProps" query="thread.myQuery">
        </cffeed>
      </cfthread>
    </cfif>
  </cfelse>
    <!--- If the user didn't fill all fields, show an error message. --->
    <h3>This example requires three feeds.<br />
    Click the Back button and try again.</h3>
    <cfabort>
  </cfif>
</cfloop>

<!--- Join the three threads. Use a 6 second time-out. --->
<cfthread action="join" name="t1,t2,t3" timeout="6000" />

<!--- Use a loop to display the results from the feeds. --->
<cfloop index="i" from="1" to="3">
  <!--- Use the cfthread scope and associative array notation to get the Thread scope dynamically. --->
  <cfset feedResult=cfthread["t#i#"]>
    <!--- Display feed information only if you got items, for example the feed must complete before the join. --->
    <cfif isDefined("feedResult.myQuery")>
      <cfoutput><h2>#feedResult.myProps.title#</h2></cfoutput>
      <cfoutput query="feedResult.myQuery">
        <p><a href="#RSSLINK#">#TITLE#</a></p>
      </cfoutput>
    </cfif>
  </cfloop>
</cfif>

<!--- The form for entering the feeds to aggregate. --->
<cfform>
  <h3>Enter three RSS Feeds</h3>
  <cfinput type="text" size="100" name="Feed1" validate="url" value="http://rss.adobe.com/events.rss?locale=en"><br />
  <cfinput type="text" size="100" name="Feed2" validate="url" value="http://weblogs.macromedia.com/dev_center/index.rdf"><br />
  <cfinput type="text" size="100" name="Feed3" validate="url" value="http://rss.adobe.com/studio.rss?locale=en"><br />
  <cfinput type="submit" name="submit">
</cfform>
**cfthrow**

**Description**

Throws a developer-specified exception, which can be caught with a `cfcatch` tag that has any of the following `type` attribute options:

- `type = "custom_type"`
- `type = "Application"`
- `type = "Any"`

**Category**

Exception handling tags, Flow-control tags

**Syntax**

```xml
<cfthrow
    message = "message"
    type = "exception type"
    detail = "detail description"
    errorCode = "error code"
    extendedInfo = "additional information"
    object = "java except object">

OR

<cfthrow
    object = #object_name#>
```

**Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

**See also**

cferror, cfrethrow, cftry, onError; Handling Errors in the Developing ColdFusion Applications

**History**

ColdFusion MX: Changed thrown exceptions: this tag can throw ColdFusion component method exceptions.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th><strong>Attribute</strong></th>
<th><strong>Optional</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>detail</td>
<td>Optional</td>
<td>Description of the event. ColdFusion appends error position to description; server uses this parameter if an error is not caught by your code.</td>
</tr>
<tr>
<td>errorCode</td>
<td>Optional</td>
<td>A custom error code that you supply.</td>
</tr>
<tr>
<td>extendedInfo</td>
<td>Optional</td>
<td>A custom error code that you supply.</td>
</tr>
<tr>
<td>message</td>
<td>Optional</td>
<td>Message that describes exception event.</td>
</tr>
<tr>
<td>object</td>
<td>Optional</td>
<td>Requires the value of the cfobject tag name attribute. Throws a Java exception from a CFML tag. This attribute is mutually exclusive with all other attributes of this tag.</td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
<td>Application</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A custom type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Application Do not enter another predefined type; types are not generated by ColdFusion applications. If you specify Application, you need not specify a type for cfcatch.</td>
</tr>
</tbody>
</table>

**Usage**

Use this tag within a **cftry** block, to throw an error. The **cfcatch** block can access accompanying information, as follows:

- **Message**, with **cfcatch.message**
- **Detail**, with **cfcatch.detail**
- **Error code**, with **cfcatch.errorcode** To get more information, use **cfcatch.tagContext**. This array shows where control switches from one page to another in the tag stack (for example, **cfinclude**, **cfmodule**).

To display the information displayed by **tagContext** variable, select the “Enable Robust Exception Information” option on the Debugging & Logging > Debug Output Settings page of the ColdFusion Administrator.
To use this tag with the object parameter, first use a cfobject tag that specifies a valid Java exception class. For example, the following cfobject tag defines an object, obj, of the exception class myException (which you must create in Java):

```
<cfobject
   type="java"
   action="create"
   class="myException"
   name="obj">
```

If your exception class has constructors that take parameters, such as a message, you can use the special init method to invoke the constructor, as in the following line. If you do not need to specify any constructor attributes, you can omit this step.

```
<cfset obj.init("You must save your work before proceeding")>
```

You can then use the cfthrow statement to throw the exception as follows:

```
<cfthrow object=#obj#>
```

For more information on using Java objects in ColdFusion, see Integrating JEE and Java Elements in CFML Applications in the Developing ColdFusion Applications.
The following example shows how to throw an exception from a component method:

```coldfusion
cfcomponent
<cffunction name="getEmp">
<cfargument name="lastName" required="yes">
<cfquery name="empQuery" datasource="cfdocexamples" >
SELECT LASTNAME, FIRSTNAME, EMAIL FROM tblEmployees WHERE LASTNAME LIKE '#arguments.lastName#' 
</cfquery>
<cfif empQuery.recordcount LT 1>
<cfthrow type="noQueryResult"
message="No results were found. Please try again.">
</cfifelse>
<cfreturn empQuery>
</cffunction>
</cfcomponent>
```

For an explanation of the example and more information, see Building and Using ColdFusion Components in the Developing ColdFusion Applications.
cftimer

Description

Displays execution time for a specified section of CFML code. ColdFusion displays the timing information along with any output produced by the timed code.

⚠️ Note

To permit this tag to execute, enable the Enable Debugging and the Timer Information options on the Debugging Settings page in the ColdFusion Administrator. Also, the IP address of the machine that runs ColdFusion must be added to the list of debugging IP addresses in the Debugging IP Addresses page if the request is sent by a remote machine. If the request is from a localhost, the IP address 127.0.0.1 must be present in the list of debugging IP addresses.

Category

Debugging tags

Syntax

```
<cftimer
    label= "text"
    type = "inline|outline|comment|debug" >
    CFML statement(s)
</cftimer>
```

⚠️ Note

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

See also

cfdump, cftrace; Debugging and Troubleshooting Applications in the Developing ColdFusion Applications

History

ColdFusion MX 7: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>Optional</td>
<td>&quot; &quot;</td>
<td>Label to display with timing information.</td>
</tr>
<tr>
<td>type</td>
<td>Optional</td>
<td>debug</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>-------</td>
<td></td>
</tr>
</tbody>
</table>

- **inline**: displays timing information inline, following the resulting HTML.
- **outline**: displays timing information and also displays a line around the output produced by the timed code. The browser must support the FIELDSET tag to display the outline.
- **comment**: displays timing information in an HTML comment in the format <!-- label: elapse d-time ms -->. The default label is `cftime`.
- **debug**: displays timing information in the debug output under the heading CFTimer Times.

**Usage**

Use this tag to determine how long it takes for a block of code to execute. You can nest `cftimer` tags. This tag is useful for debugging CFML code during application development. In production, you can leave `cftimer` tags in your code as long as you have disabled the debugging option in the ColdFusion Administrator.

**Example**

```
<!--- type="inline"> --->
<cftimer label="Query and Loop Time Inline" type="inline">
  <cfquery name="empquery" datasource="cfdocexamples">
    SELECT *
    FROM Employees
  </cfquery>

  <cfloop query="empquery">
    <cfoutput>%lastname%, %firstname%</cfoutput><br>
  </cfloop>
</cftimer>
<br>

<!--- type="outline" --->
<cftimer label="Query and CFOUTPUT Time with Outline" type="outline">
  <cfquery name="coursequery" datasource="cfdocexamples">
    SELECT *
    FROM CourseList
  </cfquery>

  <cfloop query="coursequery">
    <cfoutput>%name%</cfoutput><br>
  </cfloop>
</cftimer>
<br>
```
<cfquery>
<cfoutput query="coursequery">
<tr>
<td>#Course_ID#</td>
<td>#CorName#</td>
<td>#CorLevel#</td>
</tr>
</cfoutput>
</cfquery>
<hr>
<!--- type="comment" --->
<cfquery name="parkquery" datasource="cfdocexamples">
SELECT *
FROM Parks
</cfquery>
<p>Select View &gt; Source to see timing information</p>
<cfoutput query="parkquery">
<tr>
<td>#Parkname#</td>
</tr>
</cfoutput>
</cfquery>
<hr>
<!--- type="debug" --->
<cfquery name="deptquery" datasource="cfdocexamples">
SELECT *
FROM Departments
</cfquery>
<p>Scroll down to CFTimer Times heading to see timing information</p>
<cfoutput query="deptquery">
<tr>
<td>#Dept_ID#</td>
<td>#Dept_Name#</td>
</tr>
</cfoutput>
</cfquery>
</cftimer>
...

cftooltip

Description

Specifies tool tip text that displays when the user hovers the mouse pointer over the elements in the tag body. This tag does not require a form and is not used inside Flash forms.

Category

Display management tags

Syntax

```xml
<cftooltip
    autoDismissDelay="5000"
    hideDelay="250"
    preventOverlap="true|false"
    showDelay="200"
    sourceForTooltip="URL"
    style="CSS style specification"
    tooltip="text">
    Display tags
</cftooltip>
```

This tag must have an end tag.

⚠️ Note

You can specify this tag's attribute in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute name as structure key.

See also

cfajaximport, Using Ajax User Interface Components and Features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this tag

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoDismissDelay</td>
<td>Optional</td>
<td>5000</td>
<td>The number of milliseconds between the time when the user moves the mouse pointer over the component (and leaves it there) and when the tool tip disappears.</td>
</tr>
<tr>
<td>hideDelay</td>
<td>Optional</td>
<td>250</td>
<td>The number of milliseconds to delay between the time when the user moves the mouse pointer away from the component and when the tool tip disappears.</td>
</tr>
<tr>
<td>preventOverlap</td>
<td>Optional</td>
<td>true</td>
<td>A Boolean value specifying whether to prevent the tool tip from overlapping the component that it describes.</td>
</tr>
<tr>
<td>showDelay</td>
<td>Optional</td>
<td>200</td>
<td>The number of milliseconds to delay between the time when the user moves the mouse over the component and when the tool tip appears.</td>
</tr>
<tr>
<td>sourceForTooltip</td>
<td>Optional</td>
<td></td>
<td>The URL of a page with the tool tip contents. The page can include HTML markup to control the format, and the tip can include images. If you specify this attribute, an animated icon appears with the text &quot;Loading...&quot; while the tip is being loaded.</td>
</tr>
<tr>
<td>style</td>
<td>Optional</td>
<td></td>
<td>A CSS style specification for the tooltip. Use this attribute to set the width, text color, background color, padding, and other style properties.</td>
</tr>
</tbody>
</table>
tooltip | Optional
---|---
Tip text to display. The text can include HTML formatting. Ignored if you specify a `sourceForTooltip` attribute.

Usage

Specify a `tooltip` or a `sourceForTooltip` attribute; otherwise, this tag has no effect.

If you specify the path to a CFML page in the `sourceForTooltip` attribute, ColdFusion processes the page and uses its output in the tip text. You can therefore use CFML programming, in addition to HTML formatting, to control the contents and appearance of the tip text. You must use the `cftooltip` tag for text and simple components, such as images, not for complex Ajax components such as windows, pods, or layout areas. If you use the `cftooltip` tag with complex components, you might get unexpected behavior; for example, the tool tip might overlap window contents, even if you specify the `preventoverlap` attribute.

You can nest tool tips within the `cfinput`, `cfgrid`, and `cfform` tags, although this may result in multiple tool tips obscuring one another.

Example

The following simple example can dynamically display different tool-tip text based on the value of the `theItem` variable on the main CFML page.

The main CFML page:

```cfml
<cfoutput>
  <!--- These variables could be set dynamically --->
  <cfset theItem="left-handed & other specialty wrenches">
  <cfset theImage="lhbwrench.jpg">

  <!--- The theItem string has an ampersand, so you must URL-encode it. --->
  <cftooltip sourceForTooltip="tiptext.cfm?itemid=#URLEncodedFormat(theItem)#">
    <cfoutput>
      <b>Try this one!</b>
      <img src="#theImage#" />
    </cfoutput>
  </cftooltip>
</cfoutput>
```

The `tiptext.cfm` page could have a single CFML tag:

```cfml
<cfoutput><b> Click to find more about #URL.itemid# </b></cfoutput>
```
**cftrace**

**Description**

Displays and logs debugging data about the state of an application at the time the `cftrace` tag executes. Tracks run-time logic flow, variable values, and execution time. Displays output at the end of the request or in the debugging section at the end of the request; or, in Dreamweaver MX and later, in the Server Debug tab of the Results window. ColdFusion logs `cftrace` output to the file logs\cftrace.log, in the ColdFusion installation directory.

**Note**

To permit this tag to execute, enable debugging in the ColdFusion Administrator. Optionally, to report trace summaries, enable the Trace section.

**Category**

*Debugging tags, Variable manipulation tags*

**Syntax**

```
<cftrace
  var = "variable name"
  text = "string"
  type = "format"
  category = "string"
  inline = "yes|no"
  abort = "yes|no">
</cftrace>
```

**Note**

You can specify this tag’s attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag’s attribute names as structure keys.

**See also**

`cfdump, cferror, cfrethrow, cftry, cftimer, cftry`; *Debugging and Troubleshooting Applications* in the *Developing ColdFusion Applications*

**History**

ColdFusion MX: Added this tag.

**Attributes**

| Attribute | Req/Opt | Default | Description |
|-----------|---------|---------|-------------|-------------|
|           |         |         |             |             |
| abort | Optional | no | • yes: calls a cfabort tag when the tag is executed.  
• no |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>Optional</td>
<td></td>
<td>User-defined string that identifies trace groups.</td>
</tr>
</tbody>
</table>
| inline | Optional | no | • yes: displays trace code inline on the page in the location of the cftrace tag, in addition to the debugging information output. 
• no |
| text | Optional | | User-defined string, which can include simple variables, but not complex variables such as arrays. Outputs to the cflog tex attribute. |
| type | Optional | Information | Corresponds to the cflog type attribute; displays an appropriate icon:  
• Information  
• Warning  
• Error  
• Fatal Information |
| var | Optional | | The name of a simple or complex variable to display. Useful for displaying a temporary value, or a value that does not display on any CFM page. |

**Usage**

You cannot put application code within this tag. (This avoids problems that can occur if you disable debugging.) This tag is useful for debugging CFML code during application development. You can display cftrace tag output in the following ways:

- As a section in the debugging output
- Inline in an application page, and as a section in debugging output. If you specify inline tracing, ColdFusion flushes all output up to the cftrace tag, and displays the trace output when it encounters the tag.

The following is an example of a log file entry:
For a complex variable, ColdFusion lists the variable name and the number of elements in the object; it does not log the contents of the variable.

Example

The following example traces a FORM variable that is evaluated by a `cfif` block:

```cfc
<cftrace var="FORM.variable"
    text="doing equivalency check for FORM.variable"
    category="form_vars"
    inline="true">
    <cfif isDefined("FORM.variable") AND #FORM.variable# EQ 1>
        <h1>Congratulations, you're a winner!</h1>
    </cfif>
    <cfelse>
        <h1>Sorry, you lost!</h1>
    </cfelse>
</cftrace>
```
cftransaction

Description

For enterprise database management systems that support transaction processing, instructs the database management system to treat multiple database operations as a single transaction. Provides database commit and rollback processing. See the documentation for your database management system to determine whether it supports SQL transaction processing.

Category

Database manipulation tags

Syntax

```xml
<cftransaction
  action = "begin|commit|rollback|setsavepoint"
  isolation = "read_uncommitted|read_committed|repeatable_read"
  savepoint = "savepoint name">
</cftransaction>
```

Note

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also

cfinsert, cfprocparam, cfprocresult, cfquery, cfqueryparam, cfstoredproc, cfupdate: Commits, rollbacks, and transactions and Tags as functions and operators in Developing ColdFusion Applications

History

ColdFusion 8: Added the setsavepoint value to the action attribute. Added the savepoint attribute.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Optional</td>
<td>begin</td>
<td>• begin: The start of the block of code to execute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• commit: Commits a pending transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• rollback: Rolls back a pending transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• setsavepoint: Saves a specific state within a transaction</td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
| isolation | Optional | Isolation level, which indicates which type of read can occur during the execution of concurrent SQL transactions. The possible read actions include dirty read, in which a second SQL transaction reads a row before the first SQL transaction executes a COMMIT; non-repeatable read, in which a SQL transaction reads a row and then a second SQL transaction modifies or deletes the row and executes a COMMIT; and phantom, in which a SQL transaction reads rows that meet search criteria, a second SQL transaction then generates at least one row that meets the first transaction's search criteria, and then the first transaction repeats the search, resulting in a different result set.

- read_uncommitted: Allows dirty read, non-repeatable read, and phantom
- read_committed: Allows non-repeatable read and phantom. Does not allow dirty read.
- repeatable_read: Allows phantom. Does not allow dirty read or non-repeatable read.
- serializable: Does not allow dirty read, non-repeatable read, or phantom. |
| **savepoint** | **Optional** | **The name of the savepoint in the transaction. Setting savepoints lets you roll back portions of a transaction. For example, if your transaction includes an insert, an update, and a delete, and you set a savepoint after the update, you can roll back the transaction to exclude the delete.** |
| **nested** | **Optional** | **true** | **This attribute specifies whether the cftransaction tag can be nested inside another cftransaction tag. If the attribute value is false and there is a parent cftransaction tag, ColdFusion generates an error.** |

**Usage**

If you do not specify a value for the *action* attribute, automatic transaction processing proceeds as follows:

- If the *cfquery* operations within the transaction block complete without an error, the transaction is committed.
- If a *cfquery* tag generates an error within a *cftransaction* block, all *cfquery* operations in the transaction roll back.

If you do not specify a value for the *isolation* attribute, ColdFusion uses the default isolation level for the associated database. By using CFML error handling and the *action* attribute, however, you can explicitly control whether a transaction is committed or rolled back, based on the success or failure of the database query. In a transaction block, you can do the following:

- Commit a database transaction by nesting the `<cftransaction action = "commit"/>` tag in the block.
- Roll back a transaction by nesting the `<cftransaction action = "rollback"/>` tag in the block. (In these examples, the slash is an alternate syntax that is the equivalent of an end tag.) In a transaction block, you can write queries to more than one database, but you must commit or roll back a transaction to one database before writing a query to another. To control how the database engine performs locking during the transaction, use the *isolation* attribute. The *cftransaction* tag does not work as expected if you use the *cfthread* tag in it to make query calls.
- You can now nest *cftransaction* tags. Typically, ColdFusion 9 does not support nested transactions, but you can embed one *cftransaction* tag inside another. If you nest these tags, only the outermost *cftransaction* tag takes effect.

This feature lets you write functions that must run in a transaction without considering whether the function is called by code that is inside a *cftransaction* tag. Use a *cftransaction* tag in the function. If the calling code is in a transaction, the tag has no effect. If the calling code is not in a transaction, the tag starts the transaction. The following code shows nested transaction tags.
<cftransaction>
<cfquery name="iquery" datasource="dsn">
insert into region(regionid, regiondescription) values('111', 'YPR')
</cfquery>
<cftransaction>
<cfquery name="iquery" datasource="dsn">
update region set regiondescription = 'new' where regionid='111'
</cfquery>
</cftransaction>
</cftransaction>

⚠️ Note

In a realistic situation, the second cftransaction and cfquery can be written in a CFC that are, in turn, called by the first cftransaction and cfquery by passing the regionid value.

Example

<p>The cftransaction tag can be used to group multiple queries that use the cfquery tag into one business event. Changes to data that is requested by the queries are not committed to the datasource until all actions within the transaction block have executed successfully.</p>

<p>This a view-only example.</p>

<cftransaction>
<cfquery name='makeNewCourse' datasource='Snippets'>
INSERT INTO Courses
(Number, Descript)
VALUES
('#myNumber#', '#myDescription#')
</cfquery>

<cfquery name='insertNewCourseToList' datasource='Snippets'>
INSERT INTO CourseList
(CorNumber, CorDesc, Dept_ID, CorName, CorLevel, LastUpdate)
VALUES
('#myNumber#', '#myDescription#', '#myDepartment#', '#myDescription#', '#myCorLevel#', #Now()#)
</cfquery>
</cftransaction>

You can set savepoints at the completion of insert, update, and delete actions of a transaction. You then use error handling logic to determine whether it is necessary to roll back to a previous savepoint.

Example
<cftx>

<!---- This example performs batch processing of withdrawals ---->
<!---- from a bank account. The withdrawal amounts are stored ---->
<!---- in an array. ---->
<!---- There is a CFC named bank.cfc whose contains appear ---->
<!---- after the example. ---->

cftransaction

<!---- Get the account balance. ---->
cfinvoke component="bank" method="getBalance"
   returnvariable="getacctbalance" accountnum=1>

<cfloop index="withdrawnum" from="1" to="#ArrayLen(withdrawals)#">

<!---- Set a savepoint before making the withdrawal. ---->
cfset noxfer = "point" & #withdrawnum#
<cftransaction action = "setsavepoint" savepoint = ":noxfer#"/>

<!---- Make the withdrawal. ---->
cfinvoke component="bank" method="makewithdrawal"
   returnvariable="getacctbalance" accountnum=1
   withdrawamount="#withdrawals[withdrawnum]#">

<!---- Get the account balance. ---->
cfinvoke component="bank" method="getBalance"
   returnvariable="getacctbalance" accountnum=1>

<!---- If the balance is negative, roll back the transaction. ---->
cif getacctbalance.balance lt 0>
   cftransaction action="rollback" savepoint="#noxfer#" />
</cif>
</cfloop>
</cftransaction>

<!---- The bank.cfc contains the following:

cfcomponent>
cffunction name="getBalance" access="public" returntype="query">
   cfargument name="accountnum" type="numeric" required="yes">
   cfquery name="getacctbalance" datasource="testsqlserver">
       SELECT * FROM dbo.mybank
       WHERE accountid = #accountnum#
    </cfquery>
   cffreturn getacctbalance
</cffunction>

<cffunction name="makewithdrawal" access="public" returntype="query">
   cfargument name="accountnum" type="numeric" required="yes">
   cfargument name="withdrawamount" type="numeric" required="yes">
   cfquery name="withdrawfromacct" datasource="testsqlserver">
       UPDATE dbo.mybank SET balance = balance - #withdrawamount#
       WHERE accountid = 1
    </cfquery>
   cfinvoke method="getBalance" returnvariable="getacctbalance"
      accountnum=1>
   cffreturn getacctbalance
</cffunction>
</cfcomponent>
cftree

Description

Inserts a tree control in a form. Validates user selections. Used within a cfform tag block. Use a ColdFusion query to supply data to the tree.

Category

Forms tags

Syntax

```
<cftree
    name="name"
    align="top|left|bottom|baseline|texttop|absbottom|
        middle|absmiddle|right"
    appendKey="yes|no"
    bold="yes|no"
    border="yes|no"
    cache="yes|no"
    completePath="yes|no"
    delimiter="delimiter"
    enabled="yes|no"
    font="font"
    fontSize="size"
    format="applet|flash|html|object|xml"
    height="integer"
    highlightHref="yes|no"
    hScroll="yes|no"
    hSpace="integer"
    italic="yes|no"
    lookAndFeel="motif|windows|metal"
    message="text"
    notSupported="text">
    onBlur="ActionScript to invoke"
    onChange="ActionScript to invoke"
    onError="text"
    onFocus="Actionscript to invoke"
    onValidate="script name"
    required="yes|no"
    style= "style specification"
    tooltip="text"
    visible="yes|no"
    vScroll="yes|no"
    vSpace="integer"
    width="integer">
</cftree>
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.
See also
cfajaximport, cfapplet, cfcalendar, cfform, cfiformgroup, cfitem, cfgrid, cfinput, cfselect, cfslider, cftextarea, cftree
Using HTML trees in the Developing ColdFusion Applications

History

ColdFusion 8: Added support for Ajax based HTML trees, including the cache attribute and the html value for format attribute.

ColdFusion MX7.01: Added support for onBlur and onFocus events.

ColdFusion MX 7:

- Added the format attribute and support for generating Flash and XML and object output.
- Added enabled, onChange, style, tooltip, and visible attributes (Flash format only).

ColdFusion MX:

Changed behavior: ColdFusion renders a tree control regardless of whether there are any treeitems within it.

Attributes

⚠️ Note

In XML format, ColdFusion passes all attributes to the XML. The supplied XSLT skins do not handle or display XML format trees, but do display applet and Flash format trees.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/OptFormat</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required;All</td>
<td></td>
<td>Name for a tree control.</td>
</tr>
<tr>
<td>align</td>
<td>Optional;Applet, object</td>
<td></td>
<td>top, left, bottom, baseline, texttop, absbottom, middle, absmiddle, right</td>
</tr>
<tr>
<td>appendKey</td>
<td>Optional;All</td>
<td>yes</td>
<td>yes: if you use cftree item href attributes, ColdFusion appends a CFTREEITEMKEY query string variable with the value of the selected tree item to the cfform action URL. no: does not append the tree item value to the URL.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Default</td>
<td>Relevant Tags</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>bold</td>
<td>Displays tree control text in bold.</td>
<td>no</td>
<td>Optional; Applet, Flash, HTML</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>border</td>
<td>Displays a border around the tree control.</td>
<td>yes</td>
<td>Optional; Applet, object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cache</td>
<td>Applies only if the tree's child <code>treeitem</code> tag uses a bind expression. A Boolean value that specifies whether to get new data each time the user expands tree nodes, as follows:</td>
<td>yes</td>
<td>Optional; HTML</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>completePath</td>
<td>Starts the Form. <code>treename</code>.path variable with the root of the tree path when <code>cftree</code> is submitted.</td>
<td>no</td>
<td>Optional; Applet, HTML, object</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>delimiter</td>
<td>Optional; All</td>
<td>{{\}}</td>
<td>Character to separate elements in the Forms.tree name.path variable of the action page.</td>
</tr>
<tr>
<td>enabled</td>
<td>Optional; Flash</td>
<td>yes</td>
<td>Flash format only: Boolean value that specifies whether the control is enabled. A disabled control appears in light gray.</td>
</tr>
<tr>
<td>font</td>
<td>Optional; Applet, HTML</td>
<td></td>
<td>Font name for text in the tree control.</td>
</tr>
<tr>
<td>fontSize</td>
<td>Optional; Applet, Flash, HTML</td>
<td></td>
<td>Font size for text in the tree control, in pixels.</td>
</tr>
</tbody>
</table>
| format | Optional; All | applet | • applet: displays the tree using a Java applet in the browser.  
• flash: displays the tree using a Flash control  
• html: displays the tree uses Ajax-based HTML  
• object: returns the tree as a ColdFusion structure with the name specified by the name attribute, For details of the structure contents, see the section object format.  
• xml: generates an XML representation of the tree. In XML format forms, includes the generated XML in the form. In HTML format forms, puts the XML in a string variable with the name specified by the name attribute. |
<p>| height | Optional; Applet, Flash | 320(applet only) | Tree control height, in pixels. If you omit this attribute in Flash format, Flash automatically sizes the tree. |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| highlightHref| Optional; Applet, Object | yes     | • yes: highlights as a link the displayed value for any cftreeitem tag that specifies an href attribute.  
  • no: disables highlighting. |
| hScroll      | Optional; Applet, object | yes     | • yes: permits horizontal scrolling.  
  • no |
| hSpace       | Optional; Applet       |         | Horizontal spacing to left and right of tree control, in pixels.                                                                            |
| italic       | Optional; Applet, Flash, HTML | no      | • yes: displays tree control text in italics.  
  • no |
| label        | Optional; HTML         |         |                                                                                                                                            |
| lookAndFeel | Optional; Applet, object | windows | • motif: renders the tree in Motif style.  
  • windows: renders the tree in Windows style.  
  • metal: renders the tree in Java Swing style.  
  If the platform does not support a style option, the tag uses the default style for the platform. |
<p>| message      | Optional; Applet, HTML |         | Message to display if validation fails.                                                                                                    |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notSupported</td>
<td>Optional; Applet</td>
<td>Text to display if a page that contains a Java applet-based <code>cf:form</code> control is opened by a browser that does not support Java or has Java support disabled, for example: &quot;&lt;b&gt;Browser must support Java to view ColdFusion Java Applets&lt;/b&gt;&quot;. Default message: {&lt;b&gt;Browser must support Java to view ColdFusion Java Applets!&lt;/b&gt;}</td>
</tr>
<tr>
<td>onBlur</td>
<td>Optional; Flash</td>
<td>ActionScript to run when the tree loses focus.</td>
</tr>
<tr>
<td>onChange</td>
<td>Optional; Flash</td>
<td>ActionScript to run when the control changes due to user action. If you specify an onChange event handler, the Form scope of the ColdFusion action page does not automatically get information about selected items. The ActionScript onChange event handler must handle all changes and selections.</td>
</tr>
<tr>
<td>onError</td>
<td>Optional; Applet, HTML</td>
<td>A JavaScript function to run if validation fails.</td>
</tr>
<tr>
<td>onFocus</td>
<td>Optional; Flash</td>
<td>ActionScript to run when the tree gets focus. The JavaScript DOM form object, value of the name attribute, value that failed validation, and any error text specified by the message attribute are passed to the method.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>onValidate</strong></td>
<td>Optional; Applet, HTML</td>
<td></td>
</tr>
<tr>
<td><strong>required</strong></td>
<td>Optional; Applet, Flash, HTML</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>style</strong></td>
<td>Optional; Flash, HTML</td>
<td></td>
</tr>
<tr>
<td><strong>tooltip</strong></td>
<td>Optional; Flash</td>
<td></td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>Optional; HTML,</td>
<td></td>
</tr>
<tr>
<td><strong>visible</strong></td>
<td>Optional; Flash</td>
<td>yes</td>
</tr>
<tr>
<td><strong>vScroll</strong></td>
<td>Optional; Applet, object</td>
<td>yes</td>
</tr>
<tr>
<td><strong>vSpace</strong></td>
<td>Optional; Applet</td>
<td></td>
</tr>
</tbody>
</table>
### Usage

This tag must be in a `cfform` tag block. The applet format tree requires the client to download a Java applet. Also, if the client does not have an up-to-date Java plug-in installed, the system might also have to download an updated Java plug-in to display an applet format tree. The Flash format tree uses a Flash control, and can be embedded in an HTML format `cfform` tag. For this tag to work properly in Flash, HTML, or applet format, the browser must also be JavaScript-enabled.

**Note**

If you specify Flash format for this tag in an HTML format form, and you do not specify `height` and `width` attributes, Flash takes up more than the remaining visible area on the screen. If any other output follows the tree, including any form controls, users must scroll to see it. Therefore, if you follow a Flash tree in an HTML form with additional output, specify `height` and `width` values.

If the following conditions are true, a user's selection from query data that populates this tag's options continues to display after the user submits the form:

- The `cfform preserveData` attribute is set to "yes"
- The `cfform action` attribute posts to the same page as the form itself (this is the default), or the action page has a form that contains controls with the same names as corresponding controls on the user entry form

For more information, see the `cfform` tag entry.

### Form variables

When you select a tree item and submit the form that contains the tree, ColdFusion creates a structure with two variables in the action page Form scope. The structure name is the tree name. The following table lists the fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>The path through the tree to the selected node, in the form <code>[root]node_1_node_2_\ldots</code>. In applet format, the path includes the root node only if the <code>completePath</code> attribute is <code>true</code>. In Flash format, the path always includes the root node.</td>
</tr>
<tr>
<td>node</td>
<td>The value of the selected tree node.</td>
</tr>
</tbody>
</table>

Object format
If you specify `object` in the `format` attribute, ColdFusion returns the tree as a ColdFusion structure, and does not send the tree to the browser. You can, for example, loop over the structure to populate a menu, generate "breadcrumb" links for page navigation, or create a DHTML tree.

**Note**

If you specify an object format tree in an XML format form, ColdFusion does not generate the tree.

The structure variable name is specified by the `cftree name` attribute. The top level of the structure has two types of entries:

- Attribute settings
- A children array

**Attribute settings**

The structure has top-level entries with the values of the following `cftree` attributes:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>align</td>
<td></td>
</tr>
<tr>
<td>completePath</td>
<td></td>
</tr>
<tr>
<td>highlightHref</td>
<td></td>
</tr>
<tr>
<td>lookAndFeel</td>
<td></td>
</tr>
<tr>
<td>appendKey</td>
<td></td>
</tr>
<tr>
<td>delimiter</td>
<td></td>
</tr>
<tr>
<td>hScroll</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>bold</td>
<td></td>
</tr>
<tr>
<td>fontWeight</td>
<td></td>
</tr>
<tr>
<td>italic</td>
<td></td>
</tr>
<tr>
<td>vscroll</td>
<td></td>
</tr>
<tr>
<td>border</td>
<td></td>
</tr>
</tbody>
</table>

**Children array**

The top-level children entry is an array of items entries. Each item has the following entries:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>children</td>
<td>This item's child items; an array of item structures.</td>
</tr>
<tr>
<td>display</td>
<td>Tree item label, as determined by the <code>cftreeitem display</code> attribute.</td>
</tr>
<tr>
<td>expand</td>
<td>Whether to expand the item to display any children; value of <code>cftreeitem expand</code> attribute.</td>
</tr>
<tr>
<td>href</td>
<td>The URL to link to when the user selects the item; value of the <code>cftreeitem href</code> attribute.</td>
</tr>
<tr>
<td>img</td>
<td>The tree image icon Image to display as an icon for the tree item; value of <code>cftreeitem img</code> attribute. You can use the <code>img</code> attribute to display custom icons only in the Applet version; not in the Flash version.</td>
</tr>
<tr>
<td>imgOpen</td>
<td>Image to display when the tree item is open (expanded); value of cftreeitem imgopen attribute.</td>
</tr>
<tr>
<td>parent</td>
<td>Value of this item's parent item in the tree.</td>
</tr>
<tr>
<td>path</td>
<td>The node path from the tree root to the current element.</td>
</tr>
<tr>
<td>queryAsRoot</td>
<td>Whether the query is the root of the item; value of cftreeitem queryAsRoot attribute.</td>
</tr>
<tr>
<td>target</td>
<td>The link target, such as _blank; value of the item's cftreeitem target attribute.</td>
</tr>
<tr>
<td>value</td>
<td>The item's value, as determined by the cftreeitem value attribute.</td>
</tr>
</tbody>
</table>

**Example**

The following example creates a tree that shows available courses from the CourseList table of the cfdocexamples database, and puts each department's courses in a folder. This example is displayed in Flash and uses the Departments list to get department names.

```cfml
<cfquery name="getCourses" datasource="cfdocexamples">
    SELECT d.dept_name, c.course_id, c.CorName, c.CorLevel, c.corName ||' (' ||c.corLevel ||' ')' AS corLabel
    FROM CourseList c, Departments d
    WHERE d.Dept_ID = c.Dept_ID
    ORDER BY d.dept_Name, c.corName, c.corLevel
</cfquery>

<cfform name="studentForm" format="flash" width="400" height="450">
    <cftree name="courseTree" width="350" height="400">
        <cftreeitem
            query="getCourses"
            value="dept_name,Course_id"
            display="dept_name,CorLabel" queryAsRoot="NO" expand="yes,no">
        </cftree>
    </cfform>
</cfml>
```

The following example creates a tree that shows the basic information about all employees in an organization, organized by department. The departments are expanded to show all employees. You click the + signs to display additional information. If you click the employee name, ColdFusion links back to the same page and displays the Path and node values for the selection.
<!-- Query the datasource to get employee information. --></br>
<!-- Group the output by Department. (All fields are required in Group By clause.) --></br>
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
    SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
    FROM Employees
    GROUP BY Department, Emp_ID, FirstName, LastName, EMail, Phone</br>
</cfquery>
<html>
<body>
<h3>cftree Example</h3>

<!-- The following runs if the user clicked on a link in the tree. A complete application would use the ID for additional processing. --></br>
<cfif isdefined("Form.fieldnames")>
    <b>Selected item information</b><br>
    <cfoutput>
        <b>Path: </b>#form.Employees.Path#<br>
        <b>node: </b>#form.Employees.node#<br>
    </cfoutput>
</cfif>
<!-- Display the tree. The cftree tag must be in a cfform. --></br>
<cfform action="#cgi.script_name#" preservedata="Yes" format="Flash">
    <cftree name = "Employees" height = "400" width = "400"
        font = "Arial Narrow" italic="yes" highlightref="No" HScroll="no" VScroll="no"
        completepath="no" lookandfeel="windows" borders="No" required="yes">
        <!-- cfoutput tag with a group attribute loops over the departments. --></br>
        <cfoutput group="Department" query = "GetEmployees">
            <cftreeitem value="#Department#" parent="Employees" expand="yes">
                <!-- This cfoutput tag loops over the records for the department. The cfoutput tag does not need any attributes. --></br>
                <cfoutput>
                    <!-- Create an item for each employee in the department. Do not expand children. Each employee name links to this page, and sends the employee ID in the query string. --></br>
                    <cftreeitem value = "#LastName#, #FirstName#"
                        parent = "#Department#" expand="false" img="cd"
                        href="#cgi.script_name#?user_id=#emp_id#">
                        <!-- Each Employee entry has Id, and contact info children. --></br>
                        <cftreeitem value = "#Emp_ID#" display = "Employee ID: #Emp_ID#"
                            parent = "#LastName#, #FirstName#" img="remote">
                            <!-- Each node must be unique value, so use Emp_ID om value. --></br>
                            <cftreeitem value = "#Emp_ID#_ContactInfo" display = "Contact Information"
                                parent = "#LastName#, #FirstName#" expand = "false">
                                <!-- ContactInfo has two children --></br>
                                <cftreeitem value = "#Phone#" parent = "#Emp_ID#_ContactInfo#">
                                <cftreeitem value = "#Email#" parent = "#Emp_ID#_ContactInfo#">
                                </cfoutput>
                            </cfoutput>
                        </cfoutput>
                    </cfoutput>
                </cfoutput>
            </cftreeitem>
        </cfoutput>
        </cftree></br>
    </cfform>
</body>
</html>
cftreeitem

Description

Populates a form tree control, created with the cftree tag, with one or more elements.

Category

Forms tags

Syntax

```
<cftreeitem
  value = "text"
  bind = "bind expression"
  display = "text"
  expand = "yes|no"
  href = "URL"
  img = "filename"
  imgopen = "filename"
  parent = "parent name"
  query = "queryname"
  queryAsRoot = "yes|no"
  target = "URL target">
  OR

<cftreeitem
  bind = "bind expression">
  onbinderror = "JavaScript function name"
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

History

ColdFusion 8: Added the bind and onBindError attributes.

See also

cfapplet, cfform, cfformgroup, cfformitem, cfgrid, cfinput, cfselect, cfslider, cftextarea, cftree; Building tree controls with the cftree tag and Using HTML trees in the Developing ColdFusion Applications

Attributes

Note

In XML format, ColdFusion passes all attributes to the XML. The supplied XSLT skins do not handle or display XML format trees, but do display applet and Flash format trees.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt;Format</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Required for applet, Flash, XML.value or bind is required for HTML.</td>
<td></td>
<td>Value passed when the form containing the tree is submitted. When populating a tree with data from a cfquery, you can specify multiple columns to use in a delimited list; for example, value = &quot;dept_id,emp_id&quot;. In this case, each column generates an item that is a child of the column that precedes it in the list.</td>
</tr>
<tr>
<td>bind</td>
<td>value or bind is required for HTML</td>
<td></td>
<td>A bind expression specifying a CFC function, JavaScript function, or URL that dynamically gets all tree nodes. You can use this attribute only at the top level of the tree, and in this case, the tree can have only cftreeitem tag. If you use the bind attribute, the only other allowed attribute is onBindError. For details creating trees that using binding, see Using HTML trees in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>display</td>
<td>Optional;All</td>
<td>value</td>
<td>Tree item label. When populating a tree with data from a query, specify names in a delimited list, for example:display = &quot;dept_name,emp_name&quot;</td>
</tr>
<tr>
<td>expand</td>
<td>Optional;All</td>
<td>yes</td>
<td>• yes: expands tree to show tree item children. • no: keeps tree item collapsed.</td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<p>| <strong>href</strong> | <strong>Optional;All</strong> | **URL to link to if the user clicks the tree item. If you use a query attribute, the href attribute can specify a query column that contains URLs. If href is not a query column, the attribute text must be a URL or list of URLs. When populating a tree with data from a query, specify the URLs in a comma-delimited list, for example:{{href = &quot;http://dept_svr, http://emp_svr&quot;}} |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>img</td>
<td>Optional; Applet, HTML, object</td>
<td>Image name, filename, or file URL for tree item icon. The following values are provided:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• cd</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• computer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• element</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• folder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• floppy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• fixed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• remote</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can also specify a custom image. To do so, include path and file extension; for example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>img = &quot;../images/page1.gif&quot;</code> You can also specify a path relative to the web root. Custom images are not supported for Flash format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To specify more than one image in a tree, or an image at the second or subsequent level, use commas to separate names, corresponding to level; for example: <code>img = &quot;folder,document&quot;</code> <code>img = &quot;,document&quot;</code> (example of second level)</td>
</tr>
<tr>
<td>imgopen</td>
<td>Optional; Applet, HTML, object</td>
<td>Icon displayed with open tree item, as described for the <code>img</code> attribute.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>onBindError</td>
<td>Optional; HTML</td>
<td>The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and you specified a global error handler (by using the <code>ColdFusion.setGlobalErrorHandler</code> function), it displays the error message; otherwise a default error pop-up appears.</td>
</tr>
<tr>
<td>parent</td>
<td>Optional; All</td>
<td>Value of the tree item parent. Determines the item's placement in the tree hierarchy. If omitted, the item is placed at the tree root level, or if the <code>queryAsRoot</code> attribute is True, directly under the query.</td>
</tr>
<tr>
<td>query</td>
<td>Optional; All</td>
<td>Query name to use to populate the tree item. ColdFusion generates an item for each field value in the query column list specified by the <code>value</code> attribute. The fields in each row are hierarchically linked to the first column.</td>
</tr>
<tr>
<td>attribute</td>
<td>required</td>
<td>description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>queryAsRoot</td>
<td>Optional</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Applies only if you specify a query attribute. Defines the query as the root level for all items generated by this tag. This attribute lets you avoid creating a parent cftreeitem.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes: generates a parent (root) item for all other items generated by the tag, with the query name as its value; if you specify a parent attribute, the root item is a child of the specified parent.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no: uses the item specified by the parent attribute as the immediate parent of all items generated by this tag. If there is no parent attribute, use the query as the parent.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A string: creates a root item and uses the specified string as the item name; if you specify a parent attribute, the root item is a child of the specified parent.</td>
</tr>
<tr>
<td>target</td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Target attribute of href URL. When populating a tree with data from a query, specify targets in a comma-delimited list, for example: target = &quot;FRAME_BODY,_blank&quot;</td>
</tr>
</tbody>
</table>

Usage

For this tag to work properly, the browser must be JavaScript-enabled. This tag must be a child of a cftree tag. The cftreeitem tag has three basic formats:

- If you do not use a query or bind attribute to populate this tag, it creates a single tree item.
- If you use a query, it creates multiple items; each row of the query creates a hierarchically nested set of items with one item per column.
- If you use the bind attribute, the client side tree dynamically gets the data for the tree item’s immediate children, and creates the child items, when a tree item expands. For detailed information on using the bind a
tribute to populate an HTML format tree, see Using HTML trees in the Developing ColdFusion Applications.

Example

The following example creates a simple tree by using a single `cftreeitem` tag and a query:

```cfc
<cfform action="#cgi.script_name#">
  <cftree name="Employees" height = "400" width = "200">  
    <cftreeitem value="LastName, FirstName, Emp_ID" query="getEmployees" queryAsRoot="False">
    </cftreeitem>
  </cftree>
</cfform>
```

The following example creates a tree that shows the basic information about all employees in an organization, organized by department. The departments are expanded to show all employees. You click the + signs to display additional information. If you click the employee name, ColdFusion links back to the same page and displays the selected employee’s ID.
<!-- Query the datasource to get employee information.--->
<!-- Group the output by Department. 
(All fields are required in Group By clause.) --->
<cfquery name = "GetEmployees" dataSource = "cfdocexamples">
SELECT Emp_ID, FirstName, LastName, EMail, Phone, Department
FROM Employees
GROUP BY Department, Emp_ID, FirstName, LastName, EMail, Phone
</cfquery>
<html>
<body>
<h3>cftreeitem Example</h3>

<!-- The following runs if the user clicked on a link in the tree. 
A complete application would use the ID for additional processing. --->
<cfif isdefined("URL.user_ID")>
<cfoutput>

<!-- URL.cftreeitemkey is the selected tree item's value attribute. --->
You Requested information on #URL.cftreeitemKey#; User ID #URL.user_ID#
</cfoutput>
<br><br>
</cfif>

<!-- Display the tree. The cftree tag must be in a cfform. --->
<cfform>
<cftree name = "Employees" height = "400" width = "200" 
font = "Arial Narrow" highlighthref="No" hscroll="No">
<!-- cfoutput tag with a group attribute loops over the departments. --->
<cfoutput group="Department" query = "GetEmployees">
<cftreeitem value="#Department#" parent="Employees" expand="yes">
<!-- This cfoutput tag loops over the records for the department. 
The cfoutput tag does not need any attributes. --->
<cfoutput>

<!-- Create an item for each employee in the department. 
Do not expand children. Each employee name links to this page, 
and sends the employee ID in the query string.--->
<cftreeitem value="#LastName#, #FirstName#" 
display = "#LastName#, #FirstName#" 
parent = "#Department#" expand="no" 
href="#cgi.script_name#?user_id=#emp_id#">
<!-- Each Employee entry has ID and ContactInfo children. --->
<cftreeitem value="#Emp_ID#" display = "Employee ID: #Emp_ID#" 
parent = "#LastName#, #FirstName#">
<!-- Each node must be unique value, so use Emp_ID om value. --->
<cftreeitem value="#Emp_ID#_ContactInfo" 
display = "Contact Information" 
parent = "#LastName#, #FirstName#" expand = "No">
<!-- ContactInfo has two children. --->
<cftreeitem value="#Phone#" parent = "#Emp_ID#_ContactInfo">
<cftreeitem value="#Email#" parent = "#Emp_ID#_ContactInfo">
</cfoutput>
</cfoutput>
</cftree>
</cfform>
**cftry**

**Description**

Used with one or more `cfcatch` tags. Together, they catch and process exceptions in ColdFusion pages. Exceptions are events that disrupt the normal flow of instructions in a ColdFusion page, such as failed database operations, missing include files, and developer-specified events.

**Category**

Exception handling tags

**Syntax**

```
<cftry>
   Code that might throw an exception
   One or more cfcatch blocks
</cftry>
```

**See also**

`cfcatch, cffinally, cferror, cfrethrow, cfthrow, onError; Handling Errors` in the Developing ColdFusion Applications

**History**

ColdFusion MX: Changed `cfscript` to include `try` and `catch` statements that are equivalent to the `cftry` and `cfcatch` tags.

**Usage**

Within a `cftry` block, put the code that might throw an exception, followed by one or more `cfcatch` tags that catch and process exceptions. This tag requires an end tag.

**Example**
<h3>cftry Example</h3>
<cftry>

<!---- Note misspelled tablename "employees" as "employeeas". --->
<cfquery name = "TestQuery" dataSource = "cfdocexamples">
SELECT *
FROM EMPLOYEES
</cfquery>

<!---- <p>... other processing goes here --->
<!---- specify the type of error for which we search --->
<cfcatch type = "Database">
<!---- the message to display --->
<h3>You've Thrown a Database <b>Error</b></h3>
<cfoutput>
<!---- and the diagnostic message from the ColdFusion server --->
<p>#cfcatch.message#</p>
<p>Caught an exception, type = #CFCATCH.TYPE#</p>
The contents of the tag stack are:</p>
<cfloop index = i from = 1
to = #ArrayLen(CFCATCH.TAGCONTEXT)#>
<cfset sCurrent = CFCATCH.TAGCONTEXT[i]#>
<br="#i# #sCurrent['ID']#(#sCurrent['LINE'],#sCurrent['COLUMN']#)
#sCurrent['TEMPLATE']#
</cfloop>
</cfoutput>
</cfcatch>
</cftry>
Tags u-z

This section lists tags starting with alphabets U-Z with their description.

- cfupdate
- cfwddx
- cfwebsocket
- cfwindow
- cfxml
- cfzip
- cfzipparam
**cfupdate**

**Description**

Updates records in a data source from data in a ColdFusion form or form Scope.

**Category**

*Database manipulation tags*

**Syntax**

```
<cfupdate
dataSource = "ds_name"
tableName = "table_name"
formFields = "field_names"
password = "password"
tableOwner = "name"
tableQualifier = "qualifier"
username = "username">
```

**Note**

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

**See also**

`cfinsert, cfproccparam, cfproccresult, cfquery, cfqueryparam, cfstoredproc, cftransaction; Creating an update action page with cfupdate in the Developing ColdFusion Applications.`

**History**

ColdFusion 10: Added the clientInfo attribute. ColdFusion MX: Deprecated the connectString, dbName, dbServer, dbtype, provider, and providerDSN attributes. They do not work, and might cause an error, in releases later than ColdFusion 5.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientInfo</td>
<td>Optional</td>
<td></td>
<td>Structure containing properties of the client to be set on the database connection.</td>
</tr>
<tr>
<td>dataSource</td>
<td>Required</td>
<td></td>
<td>Name of the data source that contains the table.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Required/Optional</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>tableName</td>
<td>Required</td>
<td>Name of table to update.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- For Oracle drivers, must be uppercase.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- For Sybase driver, case sensitive; must be in same case as used when the table was created.</td>
<td></td>
</tr>
<tr>
<td>formFields</td>
<td>Optional</td>
<td>Comma-delimited list of form fields to update.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(all on form, except keys)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If a form field is not matched by a column name in the database, ColdFusion throws an error. The formFields list must include the database table primary key field, which must be present in the form. It can be hidden.</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td>Overrides the password value specified in ODBC setup.</td>
<td></td>
</tr>
<tr>
<td>tableOwner</td>
<td>Optional</td>
<td>For data sources that support table ownership (for example, SQL Server, Oracle, Sybase SQL Anywhere), the table owner.</td>
<td></td>
</tr>
<tr>
<td>tableQualifier</td>
<td>Optional</td>
<td>For data sources that support table qualifiers. The purpose of table qualifiers is as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SQL Server and Oracle: name of the database that contains the table</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Intersolv dBASE driver: directory of DBF files</td>
<td></td>
</tr>
<tr>
<td>username</td>
<td>Optional</td>
<td>Overrides username value specified in ODBC setup.</td>
<td></td>
</tr>
</tbody>
</table>

**Example**
<!--- This example lets you update a person's telephone number in the employee table. --->
<cfif isDefined("form.phone")>
  <cfupdate datasource="cfdocexamples" tablename="EMPLOYEES">
  </cfif>

<cfquery name="empTable" datasource="cfdocexamples">
  SELECT * FROM EMPLOYEES
</cfquery>

<!--- This code shows the contents of the employee table and allows you to choose a row for updating. --->
<table border="1">
  <cfoutput query="empTable">
    <tr>
      <td>#firstName#</td>
      <td>#lastName#</td>
      <td>#phone#</td>
      <td><a href="cfupdate.cfm?id=#emp_id#">Edit</a></td>
    </tr>
  </cfoutput>
</table>

<cfif isDefined("url.id")>
  <cfquery name="phoneQuery" datasource="cfdocexamples">
    SELECT * FROM employees WHERE emp_id=#url.id#
  </cfquery>
  <!--- This code displays the row to edit for update. --->
  <cfoutput query="phoneQuery">
    <form action="cfupdate.cfm" method="post">
      #phoneQuery.firstName# #phoneQuery.lastName#
      <input name="phone" type="text" value="#phone#" size="12">
      <input type="submit" value="Update">
      <input name="emp_id" type="hidden" value="#emp_id#">
    </form>
  </cfoutput>
</cfif>

Note

The cfupdate tag internally uses parameterized queries.
cfwddx

Description

Serializes and deserializes CFML data structures to the XML-based WDDX format. The WDDX is an XML vocabulary for describing complex data structures in a standard, generic way. Implementing it lets you use the HTTP protocol to such information among application server platforms, application servers, and browsers. This tag generates JavaScript statements to instantiate JavaScript objects equivalent to the contents of a WDDX packet or CFML data structure. Interoperates with Unicode.

Category

Extensibility tags

Syntax

```
<cfwddx
   action = "cfml2wddx|wddx2cfml|cfml2js|wddx2js"
   input = "inputdata"
   output = "result variable name"
   topLevelVariable = "top-level variable name for JavaScript"
   useTimeZoneInfo = "yes|no"
   validate = "yes|no"
>
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

cfcollection, cfdump, cfexecute, cfindex, cfoject, cfreport, cfsearch, ToScript; Using XML and WDDX in the Developing ColdFusion Applications

History

ColdFusion MX

- Changed column name case behavior: ColdFusion preserves the case of column names in JavaScript. (Earlier releases converted query column names to lowercase.)
- Changed encoding format support: this tag supports several encoding formats. The default encoding format is UTF-8. The tag interoperates with Unicode.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>input</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>output</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>topLevelVariable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>useTimeZoneInfo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>validate</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| **action** | Required | • `cfml2wddx`: serializes CFML to WDDX.  
• `wddx2cfml`: deserializes WDDX to CFML.  
• `cfml2js`: serializes CFML to JavaScript.  
• `wddx2js`: deserializes WDDX to JavaScript. |
| **input** | Required | A value to process. |
| **output** | Required if `action = "wddx2cfml"` | Name of variable for output. If `action = "WDDX2JS"` or "CFML2JS", and this attribute is omitted, result is output in HTML stream. |
| **topLevelVariable** | Required if `action = "wddx2js"` or "cfml2js" | Name of top-level JavaScript object created by deserialization. The object is an instance of the WddxRecordset object. |
| **useTimeZoneInfo** | Optional | yes |
| **validate** | Optional | no |
|  |  | Applies if `action = "wddx2cfml"` or "wddx2js".  
• yes: validates WDDX input with an XML parser using WDDX DTD. If parser processes input without error, packet is deserialized. Otherwise, an error is thrown.  
• no: does not perform input validation. |
Usage

ColdFusion preserves the case of column names cases in JavaScript. The `wddx2js` and `cfml2js` actions create a WddxRecordset javascript object when they encounter a RecordSet java object. The serialized JavaScript code requires a wddx.js file. This tag performs the following conversions:

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFML</td>
<td>WDDX</td>
</tr>
<tr>
<td>CFML</td>
<td>JavaScript</td>
</tr>
<tr>
<td>WDDX</td>
<td>CFML</td>
</tr>
<tr>
<td>WDDX</td>
<td>JavaScript</td>
</tr>
</tbody>
</table>

For more information, and a list of the ColdFusion array and structure functions that you can use to manage XML document objects and functions, see Using XML and WDDX in the Developing ColdFusion Applications.

⚠️ Note

The `cfwddx` tag throws an exception if you attempt to serialize a CFC or user-defined function (UDF).

Example
<!--- This example shows basic use of the cfwddx tag. --->
<html>
<body>
<!--- Create a simple query. --->
<cfquery name = "q" dataSource = "cfdocexamples">
  SELECT Message_Id, Thread_id, Username FROM messages
</cfquery>

The recordset data is:...
<cfoutput query = q>
  #Message_ID# #Thread_ID# #Username#<br>
</cfoutput>

<!--- Serialize data to WDDX format. --->
Serializing CFML data...
<cfwddx action = "cfml2wddx" input = #q# output = "wddxText">

<!--- Display WDDX XML packet. --->
Resulting WDDX packet is:
<xmp><cfoutput>#wddxText</cfoutput></xmp>

<!--- Deserialize to a variable named wddxResult. --->
Deserializing WDDX packet...
<cfwddx action = "wddx2cfml" input = #wddxText# output = "qnew">

The recordset data is:...
<cfoutput query = qnew>
  #Message_ID# #Thread_ID# #Username#<br>
</cfoutput>
**cfwebsocket**

**Description**

Lets you create the WebSocket object in your CFM template. The tag creates a reference to the WebSocket JavaScript object at the client-side.

**Category**

Internet protocol tags

**History**

ColdFusion 11: Added a new attribute, secure.

ColdFusion 10: Added this tag.

**Syntax**

```
<cfwebsocket
   name="websocketName"
   onMessage="JavaScript function name"
   onOpen="JavaScript function name"
   onClose="JavaScript function name"
   onError="JavaScript function name"
   useCfAuth=true|false
   subscribeTo="channel_list"
   secure="true|false">
```

**Attribute**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td>The name of the WebSocket object. This is the reference to the JavaScript objects that are used to call WebSocket JavaScript functions.</td>
</tr>
<tr>
<td>onMessage</td>
<td>Required</td>
<td>The JavaScript function that is called when the WebSocket receives a message from the server. The following are the two supported arguments:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• aEvent: The WebSocket event that is dispatched from the server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• aToken: Used to get data received from the server.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>onOpen</td>
<td>Optional</td>
<td>The JavaScript function that is called when the WebSocket establishes a connection.</td>
</tr>
<tr>
<td>onClose</td>
<td>Optional</td>
<td>The JavaScript function that is called when the WebSocket closes a connection.</td>
</tr>
<tr>
<td>onError</td>
<td>Optional</td>
<td>The JavaScript function that is called if there is an error while performing an action over the WebSocket connection.</td>
</tr>
<tr>
<td>usecfAuth</td>
<td>Optional</td>
<td>If set to true (default), users need not authenticate for WebSocket connection (provided they have already logged in to the application). This is the default value. If false, users have to specify the credentials for the WebSocket connection.</td>
</tr>
<tr>
<td>subscribeTo</td>
<td>Optional</td>
<td>Comma-separated list of channels to subscribe to. You can specify any or all channels set in the Application.cfc.</td>
</tr>
<tr>
<td>secure</td>
<td>Optional</td>
<td>If true, the web socket communication will happen over SSL.</td>
</tr>
</tbody>
</table>

**Example**

In the following example,

- The user is automatically subscribed to the `stocks` channel.
- You have specified the channel name in the Application.cfc.
- The example uses the default channel listener.
- In the Index.cfm, you specify the channels to which the user can automatically subscribe to (using the attribute `subscribeTo`) and also define the message handler.

```
component
{
    this.name="websocketsampleappl";
    this.wschannels=[{name="stocks"]);
}
```
Index.cfm

```javascript
<script type="text/javascript">
    function mymessagehandler(aevent, atoken)
    {
        var message = ColdFusion.JSON.encode(atoken);
        var txt=document.getElementById("myDiv");
        txt.innerHTML += message + "<br>";
    }
</script>
<cfwebsocket name="mycfwebsocketobject" onmessage="mymessagehandler"
subscribeto="stocks" >
<cfdiv id="myDiv"></cfdiv>
```

The code creates a JavaScript Web Socket object named `mycfwebsocketobject`. If the server sends a message, it calls `mymessagehandler` function. Since you have specified `subscribeTo` in the `cfwebsocket` tag, the WebSocket object automatically subscribes you to the channel `stocks`. 

© 2014 Adobe Systems Incorporated. All rights reserved.
cfwindows

Description

Creates a pop-up window in the browser. Does not create a separate browser pop-up instance.

Category

Display management tags

Syntax

```html
<cfwindow
    bodyStyle = "CSS style specification"
    center="true|false"
    closable="true|false"
    destroyOnClose = "true|false"
    draggable="true|false"
    headerStyle = "CSS style specification"
    height="number of pixels"
    initShow = "false|true"
    minHeight="number of pixels"
    minWidth="number of pixels"
    modal="true|false"
    name="string"
    onBindError = "JavaScript function name"
    refreshOnShow = "false|true"
    resizable="true|false"
    source="path"
    title="string"
    width="number of pixels"
    x="numeric pixel coordinate"
    y="numeric pixel coordinate">

    window contents

</cfwindow>
```

If you use the `source` attribute, ColdFusion ignores any tag body contents. If you do not have a tag body and omit the `</window>` end tag, close the `cfwindow` tag with the `/>` character combination.

⚠️ Note

You can specify this tag's attribute in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute name as structure key.

See also

cfajaximport, cfdiv, cflayout, cfpod, ColdFusion.Window.create, Using pop-up windows in the Developing ColdFusion Applications

History
ColdFusion 8: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodyStyle</td>
<td>Optional</td>
<td></td>
<td>A CSS style specification for the window body. As a general rule, use this attribute to set color and font styles. Using this attribute to set the height and width, for example, can result in distorted output.</td>
</tr>
<tr>
<td>center</td>
<td>Optional</td>
<td>false</td>
<td>A Boolean value that specifies whether to center the window over the browser window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If true, ColdFusion ignores the x and y attribute values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If false, and you do not specify x and y attributes, ColdFusion centers the window.</td>
</tr>
<tr>
<td>closable</td>
<td>Optional</td>
<td>true</td>
<td>A Boolean value that specifies whether the user can close the window. If true, the window has an X close icon.</td>
</tr>
<tr>
<td>destroyOnClose</td>
<td>Optional</td>
<td>false</td>
<td>If true, destroys the window when it is closed.</td>
</tr>
<tr>
<td>draggable</td>
<td>Optional</td>
<td>true</td>
<td>A Boolean value that specifies whether the user can drag the window. To drag the window, click the mouse on the title bar and hold the button down while dragging. If the window does not have a title, users cannot drag it.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>headerStyle</td>
<td>Optional</td>
<td></td>
<td>A CSS style specification for the window header. As a general rule, use this attribute to set color and font styles. Using this attribute to set the height and width, for example, can result in distorted output.</td>
</tr>
<tr>
<td>height</td>
<td>Optional</td>
<td>300</td>
<td>Height of the window in pixels. If you specify a value greater than the available space, the window occupies the available space and the resize handles do not appear.</td>
</tr>
<tr>
<td>initShow</td>
<td>Optional</td>
<td>false</td>
<td>A Boolean value that specifies whether to display the window when the containing page first appears. If this value is false, use the ColdFusion.Window.show JavaScript function to display the window.</td>
</tr>
<tr>
<td>minHeight</td>
<td>Optional</td>
<td>0</td>
<td>The minimum height, in pixels, to which users can resize the window.</td>
</tr>
<tr>
<td>minWidth</td>
<td>Optional</td>
<td>0</td>
<td>The minimum width, in pixels, to which users can resize the window.</td>
</tr>
<tr>
<td>modal</td>
<td>Optional</td>
<td>false</td>
<td>A Boolean value that specifies whether the window is modal, that is, whether the user can interact with the main window while this window is displayed. If true, the user cannot interact with the main window.</td>
</tr>
<tr>
<td>attribute</td>
<td>type</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Optional</td>
<td>The name of the window. Must be unique on the pages. This attribute is required to interact with the window, including to dynamically show or hide it.</td>
<td></td>
</tr>
<tr>
<td>onBindError</td>
<td>Optional</td>
<td>The name of a JavaScript function to execute if evaluating a bind expression results in an error. The function must take two attributes: an HTTP status code and a message. If you omit this attribute, and specified a global error handler (by using the <code>ColdFusion.setGlobalErrorHandler</code> function), it displays the error message; otherwise a default error pop-up appears.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>overflow</td>
<td>Optional</td>
<td>auto</td>
<td>Specifies how to display child content whose size would cause the control to overflow the window boundaries. The following values are valid:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>auto</strong>-Show scroll bars when necessary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>hidden</strong>-Do not allow access to overflowing content.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>scroll</strong>-Always show horizontal and vertical scroll bars, even if they are not needed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>visible</strong>-Content can display outside the bounds of the window.</td>
</tr>
<tr>
<td>Note: In Internet Explorer, windows with the <strong>visible</strong> setting expand to fit the size of the contents, rather than having the contents extend beyond the layout area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>refreshOnShow</td>
<td>Optional</td>
<td>false</td>
<td>• <strong>true</strong>-Refresh the contents of the window by running the <strong>source</strong> bind expression whenever the window shows (for example, by calling the <code>ColdFusion.Window.show</code> Java Script function), in addition to when bind events occur</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>false</strong>-Refresh the window only when the bind expression is triggered by its bind event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To use this attribute, you must also specify a <strong>source</strong> attribute.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>resizable</td>
<td>Optional</td>
<td>true</td>
<td>A Boolean value specifying whether the user can resize the window.</td>
</tr>
<tr>
<td>source</td>
<td>Optional</td>
<td></td>
<td>A URL that returns the window contents. This attribute can use URL parameters to pass data to the page. ColdFusion uses standard page path resolution rules to locate the page. You can use a bind expression in this attribute; for more information see Usage. <strong>Note:</strong> If a CFML page specified in this attribute contains tags that use Ajax features, such as cf, cfgrid, and cfpod, you must use a cfajaximport tag on the page with the cfwindow tag. For more information, see <a href="#">cfajaximport</a>.</td>
</tr>
<tr>
<td>title</td>
<td>Optional</td>
<td></td>
<td>The text to display in the window's title bar. You can use HTML mark-up to control the title appearance; for example, to show the text in red italic font.</td>
</tr>
<tr>
<td>width</td>
<td>Optional</td>
<td>500</td>
<td>Width of the window in pixels. If you specify a value greater than the available space, the window occupies the available space and the resize handles do not appear.</td>
</tr>
</tbody>
</table>
| x | Optional | The X (horizontal) coordinate of the upper-left corner of the window, relative to the browser window. ColdFusion ignores this attribute if the `center` attribute value is `true` and if you do not set the `y` attribute value.

| y | Optional | The Y (vertical) coordinate of the upper-left corner of the window, relative to the browser window. ColdFusion ignores this attribute if the `center` attribute value is `true` and if you do not set the `x` attribute value.

**Usage**

You cannot use this tag in a form or as a child of a `cflayout`, `cflayoutarea` tag. Define the `cfwindow` tag on the page that displays it (or a page that is included by using the `cfinclude` tag). So, you cannot use the `cfwindow` tag on a page that is specified by a `cfmenuitem` `http` attribute, `cfdiv` `bind` attribute, or `cflayoutarea` or `cfpod` `source` attribute. Instead, for example, you can display a window when a user clicks a menu item by defining the window on the same page as your menu and using a JavaScript function in the `cfmenuitem` `http` attribute to call the window's `show` function. The `cfwindow` tag uses its `source` attribute to get its contents from another page.

You can use a `source` attribute or a tag body to specify the window contents; if you specify both, ColdFusion uses the contents specified by the `source` attribute and ignores the tag body. If you use a `source` attribute, an animated icon and the text "Loading..." appears while the contents is being fetched.

If the `source` attribute specifies a page that defines JavaScript functions, the function definitions on that page must have the following format:

```javascript
functionName = function(arguments) {function body}
```

Function definitions that use the following format may not work:

```javascript
function functionName (arguments) {function body}
```

However, Adobe recommends that you include all custom JavaScript in external JavaScript files and import them on the application’s main page, and not write them inline in code that you get by using the `source` attribute. Imported pages do not have this function definition format restriction.
If you use the `source` attribute, you can use a `bind expression` to include form field values or other form control attributes as part of the source specification. You can bind to HTML format form controls only. For detailed information on using bind expressions, see Binding data to form fields in the Developing ColdFusion Applications.

**JavaScript functions**

You can use the following JavaScript functions to manage an HTML format window:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ColdFusion.Window.create</code></td>
<td>Creates a window without using a <code>cfwindow</code> tag.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.getWindowObject</code></td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified HTML format <code>cfwindow</code> control.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.hide</code></td>
<td>Closes a window.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.onHide</code></td>
<td>Specifies a function to run each time a specific window closes.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.onShow</code></td>
<td>Specifies a function to run each time a specific window opens.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.show</code></td>
<td>Opens a window.</td>
</tr>
</tbody>
</table>

**Example**

The following example shows several features of the `cfwindow` tag and dynamic binding of the `cfwindow` tag `source` attribute to form controls. It shows how you can use `x` and `y` attributes to position the windows and how several attributes, such as `closable` and `resizable` affect the window appearance. It also shows how you can use bind expressions to dynamically update window contents when form control values change, including different ways to trigger updating the window contents.

```html
<html>
<head>
</head>
<body>
    <cfform name="myform">
        <cfinput type="hidden" name="hiddentext" value="This is hidden text from the main page">
        <cfinput name="text1"><br />
        Click the mouse on the control to show its text in window 1.
        <cfinput name="text1"><br />

        Click the button to show the input control text in window 2.
        <cfinput name="text2">
        <cfinput type="button" name="mybutton" value="Show Text" onclick="javascript:ColdFusion.Window.show('mywindow2')"><br />

        Click the Checkbox to change and show its status in window 3
        <cfinput name="check1" type="checkbox"><br />
```
Click the button to open a window containing the page specified by the input control.
<cfinput name="text3" value="windowsource.cfm">
<cfinput type="button" name="mybutton3" value="Open Window" onclick="javascript:ColdFusion.Window.show('mywindow4')">
</cfform>

<!---- This window shows initially and cannot be closed, dragged, or resized.
The value of the text URL parameter, and therefore, the contents of the text displayed in the window changes each time the user clicks the mouse over the text1 control. --->
<cfwindow x="0" y="100" width="200" height="150"
name="mywindow" title="My First Window"
closable="false" draggable="false" resizable="false" initshow="true"
source="windowsource.cfm?text={myform:text1@mousedown}" />

<!---- This window shows initially and cannot be dragged, or resized, but can be closed.
The text URL parameter represents the text2 input control value. --->
<cfwindow x="0" y="250" width="200" height="150"
name="mywindow2" title="My Second Window"
initshow="true" draggable="false" resizable="false"
source="windowsource.cfm?text={myform:text2}" />

<!---- This window shows initially and cannot be resized, but can be dragged or closed.
The value of the text URL parameter, and therefore, Boolean value displayed in the window changes each time the user clicks the mouse in the check1 control to change the check box status. The bind expression binds to the check box checked attribute; it specifies a click event because Internet Explorer does not generate a change event when the user clicks the box. --->
<cfwindow x="0" y="400" width="200" height="150"
name="mywindow3" title="My Third Window"
initshow="true" resizable="false"
source="windowsource.cfm?text=<b>Checkbox: </b>{myform:check1.checked@click}" />

<!---- This window does not display initially.
The Open Window button control opens it. It can be closed, dragged, and resized.
The value text URL parameter represents the value of a hidden text field. --->
<cfwindow x="210" y="100" width="500" height="480"
name="mywindow4" minHeight="400" minWidth="400"
title="My Fourth Window" initshow="false"
source="{myform:text3}?text={myform:hiddentext}" />
The windowsource.cfm page that the cfwindow tag source attributes specify to display in the windows contains the following code:

```html
<h3>Main page input:</h3>
<cfoutput>
#url.text#
</cfoutput>
```
cfxml

Description

Creates a ColdFusion XML document object that contains the markup in the tag body. This tag can include XML and CFML tags. ColdFusion processes the CFML code in the tag body, and then assigns the resulting text to an XML document object variable, which is always stored in Unicode.

Category

Extensibility tags

Syntax

```xml
<cfxml
  variable="xmlVarName"
  caseSensitive="yes|no">
```

Note

You can specify this tag's attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag's attribute names as structure keys.

See also

IsXmlDoc, IsXmlElem, IsXmlRoot, ToString, XmlChildPos, XmlNew, XmlParse, XmlSearch, XmlTransform: Using XML and WDDX in the Developing ColdFusion Applications

History

ColdFusion MX 7: Added support for using an XML declaration at the start of the text.

ColdFusion MX: Added this tag.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable</td>
<td></td>
<td></td>
<td>Name of the document object.</td>
</tr>
</tbody>
</table>
| caseSensitive | Optional| no      | • yes: maintains the case of document elements and attributes.  
|               |         |         | • no                                  |

Usage

If your XML object is case sensitive, you cannot use dot notation to reference an element or attribute name. Use the name in associative array (bracket) notation, or a reference that does not use the case-sensitive name (such as xmlChildren1) instead. In the following code, the first line works with a case-sensitive XML object. The second and
third lines cause errors:

```csharp
MyDoc.XmlRoot.XmlAttributes["Version"] = "12b";
MyDoc.XmlRoot.XmlAttributes.Version = "12b";
MyDoc.MyRoot.XmlAttributes["Version"] = "12b";
```

Use the XmlFormat function to escape special characters such as &, > and <. To convert an XML document object back into a string, use the `ToString` function, at which time ColdFusion automatically prepends the `<?xml version="1.0" encoding="UTF-8"?>` XML declaration. To change the declaration to specify another encoding, use the `Replace` function. To specify the encoding of the text that is returned to a browser or other application, use the `cfcontent` tag. The following example illustrates this process:

```csharp
<!--- <cfprocessingdirective suppresswhitespace="Yes">-----
<cfprocessingdirective suppresswhitespace="Yes">
<cfcontent type="text/xml; charset=utf-16">
<cfxml variable="xmlObject">
<breakfast_menu>
<food>
  <name quantity="50">Belgian Waffles</name>
  <description>Our famous Belgian Waffles</description>
</food>
</breakfast_menu>
</cfxml>
</cfprocessingdirective>

<cfprocessingdirective suppresswhitespace="Yes">
<cfcontent type="text/xml; charset=utf-16">
<cfxml variable="xmlObject">
<breakfast_menu>
<food>
  <name quantity="50">Belgian Waffles</name>
  <description>Our famous Belgian Waffles</description>
</food>
</breakfast_menu>
</cfxml>
</cfprocessingdirective>

<cfoutput>#mynewvar#</cfoutput>
```

The `cfprocessingdirective` tag prevents ColdFusion from putting white space characters in front of the XML declaration.

**Example**

This following example creates a document object whose root element is MyDoc. The object includes text that displays the value of the ColdFusion variable `testVar`. The code creates four nested child elements, which are generated by an indexed `cfloop` tag. The `cfdump` tag displays the XML document object.
<cfset testVar = True>
<cfxml variable="MyDoc">
  <?xml version='1.0' encoding='utf-8' ?>
  <MyDoc>
    <cfif testVar IS True>
      <cfoutput>The value of testVar is True.</cfoutput>
      <cfelse>
        <cfoutput>The value of testVar is False.</cfoutput>
      </cfif>
    <cfloop index = "LoopCount" from = "1" to = "4">
      <childNode>
        This is Child node <cfoutput>#LoopCount#.</cfoutput>
      </childNode>
    </cfloop>
  </MyDoc>
</cfxml>
<cfdump var=#MyDoc#>
**cfzip**

**Description**

Manipulates ZIP and Java Archive (JAR) files. In addition to the basic zip and unzip functions, use the `cfzip` tag to delete entries from an archive, filter files, read files in binary format, list the contents of an archive, and specify an entry path used in an executable JAR file.

**History**

ColdFusion 11: Added 2 new attributes, password and encryptionalgorithm.

ColdFusion 8: Added this tag.

**Category**

*File management tags*

**Syntax**

```xml
delete
cfzip
  required
  action = "delete"
  file = "absolute pathname"
  optional
  entrypath = "full pathname"
  filter = "file filter"
  recurse = "yes|no">

list
cfzip
  required
  action = "list"
  file = "absolute pathname"
  name = "recordset name"
  optional
  filter = "file filter"
  recurse = "yes|no"
  showDirectory= "yes|no">

read
cfzip
  required
  action = "read"
  entrypath = "full pathname"
  file = "absolute pathname"
  variable = "variable name"
  optional
  password = "password string"
  charset = "encoding type">

readBinary
cfzip
  required
  action = "readBinary"
  entrypath = "full pathname"```
file = "absolute pathname"
variable = "variable name"
    optional
    password = "password string">

unzip
<cfzip
    required
    action = "unzip"
    destination = "destination directory"
    file = "absolute pathname"
    optional
    password = "password string"
    entrypath = "full pathname"
    filter = "file filter"
    overwrite = "yes|no"
    recurse = "yes|no"
    storePath = "yes|no">

zip
<cfzip
    required
    file = "absolute pathname"
    One of the following:
    source = "source directory"
    <cfzipparam source = "source directory" ...>
    optional
    encryptionAlgorithm = "standard|AES-128|AES-256"
    password = "password string"
    action = "zip"
    filter = "file filter"
    overwrite = "yes|no"
    prefix = "string"
recurse = "yes|no"
storePath = "yes|no">

⚠️ **Note**

You can specify this tag's attributes in an `attributeCollection` attribute whose value is a structure. Specify the structure name in the `attributeCollection` attribute and use the tag's attribute names as structure keys.

---

See also

*cfzipparam*

---

### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Action</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| action    | N/A    | Optional| zip     | Action to take. Must be one of the following:  
- delete  
- list  
- read  
- readBinary  
- unzip  
- zip  
If you do not specify an action, ColdFusion applies the default action, zip. |
| charset   | read   | Optional| default encoding of the host machine | Character set used to translate the ZIP or JAR entry into a text string. Examples of character sets:  
- JIS  
- RFC1345  
- UTF-16 |
| destination| unzip  | Required|         | Destination directory where the ZIP or JAR file is extracted. |
| entryPath | deletereadreadBinaryunzip | Optional|         | Pathname on which the action is performed. |
| file | deletelist | Required | Absolute pathname of the file on which the action is performed; for example, the full pathname of the ZIP file: \texttt{c:\temp\log.zip}. If you do not specify the full pathname (for example, \\texttt{file= "log.zip"}), ColdFusion creates the file in a temporary directory. You can use the \texttt{GetTempDirectory} function to access the ZIP or JAR file.

| filter | deletelist | Optional | File filter applied to the action. The action applies to all files in the specified pathname that match the filter.

| name | list | Required | Record set name in which the result of the list action is stored. The record set columns are the following:
| **name**: Filename of the entry in the JAR file. For example, if the entry is help/docs/index.htm, the name is index.htm. |
| **directory**: Directory containing the entry. For the preceding example, the directory is help/docs. You can obtain the full entry name by concatenating directory and name. If an entry is at the root level, the directory is empty (" "). |
| **size**: Uncompressed size of the entry, in bytes. |
| **compressedSize**: Compressed size of the entry, in bytes. |
| **type**: Type of entry (directory or file). |
| **dateLastModified**: Last modified date of the entry, cfdate object. |
| **comment**: Any comment, if present, for the entry. |
| **crc**: Crc-32 checksum of the uncompressed entry data. |
**overwrite**  |  **unzip**  |  Optional  |  no  |  unzip: Specifies whether to overwrite the extracted files:

- *yes:* If the extracted file exists at the destination specified, the file is overwritten.
- *no:* If the extracted file exists at the destination specified, the file is not overwritten and that entry is not extracted. The remaining entries are extracted.

**zip:** Specifies whether to overwrite the contents of a ZIP or JAR file:

- *yes:* Overwrites all of the content in the ZIP or JAR file if it exists.
- *no:* Updates existing entries and adds new entries to the ZIP or JAR file if it exists.

**prefix**  |  **zip**  |  Optional  |  String added as a prefix to the ZIP or JAR entry. The string is the name of a subdirectory in which the entries are added.
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recurse</td>
<td>deletelistunzipzip</td>
<td>Optional</td>
<td>yes</td>
<td>Specifies whether the action applies to subdirectories:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• yes: Includes subdirectories.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• no: Does not include subdirectories.</td>
</tr>
<tr>
<td>showDirectory</td>
<td>list</td>
<td>Optional</td>
<td>no</td>
<td>Specifies whether to show the directory structure:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• yes: Lists the directories.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• no: Does not list directories.</td>
</tr>
<tr>
<td>source</td>
<td>zip</td>
<td>Required (see description)</td>
<td></td>
<td>Source directory to be zipped. Not required if the cfzi pparam tag is specified.</td>
</tr>
<tr>
<td>variable</td>
<td>readreadBinary</td>
<td>Required</td>
<td>Variable in which the content is stored.</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>------------</td>
<td>-------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>zip</td>
<td>Optional</td>
<td>The password to be used for the archive.</td>
<td></td>
</tr>
<tr>
<td>encryptionAlgorithm</td>
<td>zip</td>
<td>Optional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-----</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The encryption algorithm to use for compression. The following options are valid:

- AES-256
- AES-128
- Standard

If the attributes are specified in the CFZip tag, they can be inherited or overridden by the corresponding attributes in the CFZipparam tag.

If only password is provided, the default encryptionAlgorithm will be AES 256-bit.

If used with a directory, all files in that directory will be protected with the same password/encryption.

### Usage

Use the `cfzip` tag to zip and unzip files and manipulate existing ZIP or JAR files in ColdFusion. You can use the `cfzip` tag independently or with one or more `cfzipparam` tags to manipulate multiple files or directories. The `cfzip` tag is the parent tag of the `cfzipparam` tag.

The ZIP format is the standard format for file archiving and compression. The JAR format is based on the ZIP format. JAR files are platform-independent.

⚠️ **Note**

The `cfzip` tag does not create directories. If you specify a directory that does not exist, ColdFusion generates an error.

Use the following syntax to specify an in-memory file or directory in any attribute that takes a path. In-memory files are not written to disk and speed processing of transient data.

```plaintext
ram:///filepath
```
The filepath can include multiple directories, for example `ram:///petStore/images/dogImages.zip`. You must create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications.

**delete action**

Use the `delete` action to delete entries from a ZIP or JAR file.

```cftags
<!--- This example shows how to delete all the properties in a JAR file. --->
<cfzip file="e:\work\source.jar" action="delete" filter="*.properties, *.props">
<!--- This example shows how to delete all of the entries in a ZIP file with a JPG, GIF, or PNG extension, including entries in subdirectories. --->
<cfzip file="c:\myApp\images.zip" action="delete" filter="*.jpg, *.gif, *.png" recurse="yes">
<!--- This example shows how to delete the "images" subdirectory (and its contents) from the "myApp.zip" file. --->
<cfzip action="delete" file="c:\myApp.zip" entrypath="images">
<!--- This example shows how to delete all Java source entries in the "work/source" directory and images (*.gif, *.jpg, *.jpeg) from a JAR file. --->
<cfzip file="C:\downloads\source.jar" action="delete">
  <cfzipparam entrypath="work/source" filter="*.java">
    <cfzipparam filter="*.gif,*.jpg,*.jpeg">
  </cfzip>
</cfzip>
```

**list action**

Use the `list` action to list the entries of a ZIP or JAR file. The following table shows the types of information you can retrieve for entries in the archive:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment</td>
<td>Text string description saved with the entry source.</td>
</tr>
<tr>
<td>compressedSize</td>
<td>Compressed size of the entry in bytes.</td>
</tr>
<tr>
<td>crc</td>
<td>Checksum for the entry source.</td>
</tr>
<tr>
<td>dateLastModified</td>
<td>Date and time when the source was last modified.</td>
</tr>
<tr>
<td>directory</td>
<td>Name of the directory where the entry is stored.</td>
</tr>
<tr>
<td>name</td>
<td>Entry pathname.</td>
</tr>
<tr>
<td>size</td>
<td>Uncompressed size of the entry source in bytes.</td>
</tr>
<tr>
<td>type</td>
<td>Source type for the entry, for example, file.</td>
</tr>
</tbody>
</table>

You can use the `cfdump` tag to list all of the information in a ZIP or JAR file, as the following example shows:
You can use the **cfoutput** tag to list individual fields for the entries in an archive, as the following example shows:

```cfml
<cfzip file="c:\zipTest\Test.zip" action="list" name="entry">
<cfoutput>
<table>
<tr>
<td><b>Entry Name:</b> #entry.name#</td>
<td><b>Last Modified Date:</b> #dateFormat(entry.dateLastModified),#timeFormat(entry.dateLastModified)#</td>
<td><b>Size (uncompressed):</b> #numberFormat(entry.size/1000)# KB</td>
</tr>
</cfoutput>
</table>
</cfzip>
```

**read action**

Use the **read** action to read the content of the ZIP or JAR file entry in human-readable format. The **read** action uses the **charset** value to create the string.

```cfml
<cfzip action="read" file="/home/sam/work/util.jar" entrypath="info.txt" variable="text">
```

**readBinary action**

Use the **readBinary** action to read the content of a ZIP or JAR file in binary format.

```cfml
<cfzip file="c:\work\instr.zip" action="readBinary" entryPath="com/test/abc.jpg" variable="xyz">
<cfzip file="c:\work\copy_instr.zip">
<cfzipparam entryPath="com/test/xyz.jpg" content="#xyz#">
</cfzip>
```

**unzip action**

Use the **unzip** action to extract the entries from a ZIP or JAR file.
### Example

The following example shows how to zip image files chosen from a form and e-mail the ZIP file to the person requesting the images.

The first ColdFusion page populates a pop-up menu with the names of artists generated from a database query:
<!---- Create a query to extract artist names from the cfartgallery database. --->
<cfquery name="artist" datasource="cfartgallery">
  SELECT FIRSTNAME || ' ' || LASTNAME AS FULLNAME, ARTISTS.ARTISTID
  FROM ARTISTS
</cfquery>

<!---- Create a form that lists the artists generated by the query. --->
<cfform action="zipArt_action.cfm" method="post">
  <h3>Choose an Artist</h3>
  <p>Please choose an artist:</p>
  <cfselect name="artistName" query="artist" display="FULLNAME" value="ARTISTID"
     required="yes" multiple="no" size="8">
  </cfselect>
  <br/>
  <cfinput type="submit" name="submit" value="OK">
</cfform>

The first action page displays the images by the selected artist, zips the files, and writes the ZIP file to a temporary directory. Also, it includes a form to e-mail the ZIP file:
<!--- Create a query to extract artwork for the selected artist from the cfartgallery database. --->
<cfquery name="artwork" datasource="cfartgallery">
    SELECT FIRSTNAME, LASTNAME, LARGEIMAGE
    FROM ARTISTS, ART
    WHERE ARTISTS.ARTISTID = ART.ARTISTID
    AND ARTISTS.ARTISTID = <cfqueryparam value="#form.artistName#">
    ORDER BY ARTNAME
</cfquery>

<p>You have chosen the work of #artwork.FirstName# #artwork.LastName#.</p>
<cfset thisDir = ExpandPath(".")>
<cfset imgDir = ExpandPath("..")>
</cfoutput>
<cfset xctr = 1>
<table border="0" cellpadding="15" cellspacing="0" bgcolor="#FFFFFF">
<cfoutput query="artwork">
<cfif xctr mod 3 eq 1>
    <tr>
</cfif>
    <!--- Use the IsImageFile function to verify that the image files extracted from the database are valid. Use the ImageNew function to create a ColdFusion image from valid image files. --->
    <cfif IsImageFile("#imgdir#/cfdocs/images/artgallery/
        #artwork.largeImage#")>
        <cfset myImage=ImageNew(#imgdir#/cfdocs/images/artgallery/
            #artwork.largeImage#)>
        <td valign="top" align="center" width="200">
            <cfset xctr = xctr + 1>
            <img src="#imgdir#/cfdocs/images/artgallery/#artwork.largeImage#"/>
        </td>
</cfoutput>
</cfif>
</cfoutput>
</tr>
</table>
<h3>Mail the ZIP File</h3>
<p>Please enter your e-mail address so we can send you the ZIP file as an attachment.</p>
<cfform action = "zipArt_action2.cfm" method="post">
    Your e-mail address: <cfinput type = "Text" name = "MailTo">
    <!--- Specify the required field. --->
    <cfinput type = "hidden" name = "MailTo_required" value = "You must enter your email address">
    <cfinput type="hidden" name="zipPath" value="#thisDir#/artwork.#artwork.lastname#.zip">
</cfform>
<h3>Mail the ZIP file</h3>
<p>Your file has been mailed to you.</p>
<cfset eMail="#form.MailTo#"
<cfset zipPath="#form.zipPath#"
<cfmail from="coldfusion@adobe.com" to="#eMail#" subject="see zipped attachment">
  The images you requested are enclosed in a ZIP file.
  <cfmailparam file="#zipPath#">
</cfmail>
cfzipparam

Description

Provides additional information to the cfzip tag.

The cfzipparam tag is always a child tag of the cfzip tag.

History

ColdFusion 11: Added 2 new attributes, password and encryptionalgorithm.

ColdFusion 8: Added this tag.

Category

File management tags

Syntax

```html
<cfzip ..>
  <cfzipparam
    charset = "encoding type"
    content = "variable name"
    entryPath = "full pathname"
    filter = "file filter"
    prefix = "string"
    recurse = "yes|no"
    source = "source directory"
    optional
      encryptionAlgorithm = "standard|AES-128|AES-256"
      password = "password string">
</cfzip>
```

You can specify this tag’s attributes in an attributeCollection attribute whose value is a structure. Specify the structure name in the attributeCollection attribute and use the tag’s attribute names as structure keys.

See also

cfzip

Attributes

| Attribute   | Req/Opt | Default | Description |
|-------------|---------|---------|-------------|-------------|
|             |         |         |             |             |
| **charset** | Optional | default encoding of the host machine | Converts string content into binary data before putting it into a ZIP or JAR file. Used only when `cfzip action="zip"` and the `cfzipparam content` is a string.

Examples of character sets:
- JIS
- RFC1345
- UTF-16 |
| **content** | Optional | Content written to the ZIP or JAR entry. Used only when `cfzip action="zip"`.

Valid content data types are `binary` and `string`. If you specify the `content` attribute, specify the `entrypath` attribute. |
| **entryPath** | Optional | Pathname used:
- For `cfzip action="zip"`, it is the entry path used. This is valid only when the `source` is a file. The entry path creates a subdirectory in the ZIP or JAR file.
- For `cfzip action="unzip"`, it is the pathname to unzip.
- For `cfzip action="delete "`, it is the pathname to delete from the ZIP or JAR file. |
<table>
<thead>
<tr>
<th>Filter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td></td>
<td>File filter applied to the action. For example, for the <code>zip</code> action, all the files in the source directory that match the filter are zipped.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td></td>
<td>String added as a prefix to the ZIP or JAR entry. Used only when <code>cfzip</code> action=&quot;zip&quot;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recurse</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recurse</td>
<td></td>
<td>Include the directory to be zipped, unzipped, or deleted, as specified by the <code>cfzip</code> parent tag.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td></td>
<td>Source directory or file. Used only when <code>cfzip</code> action=&quot;zip&quot;. Specified files are added to the ZIP or JAR file:</td>
</tr>
</tbody>
</table>

- If you specify `source` attribute for the `cfzip` tag, the `cfzip` parameter `source` is relative to it.
- If you do not specify a `source` attribute for the `cfzip` tag, the `cfzip` parameter `source` must be an absolute pathname.
| encryptionAlgorithm | Optional | The encryption algorithm to use for compression. The following options are valid:
| | | • AES-256
| | | • AES-128
| | | • Standard
| | | If the attributes are specified in the CFZip tag, they can be inherited or overridden by the corresponding attributes in the CFZipparam tag.
| | | If only password is provided, the default encryptionAlgorithm will be AES 256-bit.
| | | If used with a directory, all files in that directory will be protected with the same password/encryption.
| password | read readBinary zip unzip | Required, if encryptionAlgorithm is specified | The password to protect the archive.

Usage

Use the `cfzipparam` tag with the `cfzip` tag to zip, extract, or delete multiple files or directories. For example, to zip multiple directories, specify `acfzipparam` tag for each source directory.

Example

Example 1

```cftags
<!--- This example shows how to zip class files from one subdirectory and class and property files from another directory into a JAR file. --->
<cfzip file="c:\util.jar" source="c:\myproj\classes">
    <cfzipparam source="com\abc\util" filter="*.class">
        <cfzipparam source="com\abc\io" filter="*.class, *.properties">
    </cfzipparam>
</cfzip>
```

Example 2
Example 3

<!-- This example shows how to insert the string format for a programmatically generated XML file into a ZIP file. -->
<!---- Create a variable that specifies a time-out period. ---->
<cfset myDoc="<system-config><timeout>1500</timeout><pool-max-size>30</pool-max-size></system-config>">

<!-- Zip the file. -->
<cfzip file="e:\work\test.zip" action="zip">
  <cfzipparam source="c:\src\Manifest.MF" entrypath="META-INF\MANIFEST">
  <cfzipparam content="#myDoc#" entrypath="system-config.xml">
</cfzip>

Example 4

<!-- This example shows how to update a JAR file with a new version of the file and add some new files to the JAR file. -->
<cfzip file="e:\work\admin.jar">
  <cfzipparam source="c:\src\classes" recurse="yes" filter="*.class,*.properties">
    <cfzipparam source="c:\src\Manifest.MF" entrypath="META-INF\MANIFEST">
  </cfzip>
</cfzip>

Example 5

The following example shows how to zip multiple image files chosen from a form and e-mail the ZIP file to the person requesting the images.

The first ColdFusion page populates a pop-up menu with the names of artists generated from a database query:
<!---- The following code creates a form for selecting images. --->
<h3>Select the images</h3>
<p>Please choose the images you would like sent to you.</p>
<!---- Create the ColdFusion form to select the images. --->
<table>
<cfform action="zip2_action.cfm" method="post"
enctype="multipart/form-data">
<tr>
<td><img src="../cfdocs/images/artgallery/elecia01.jpg"/><br/>
<cfinput type="checkbox" name="ck1" Value=1>Cube</td>
<td><img src="../cfdocs/images/artgallery/elecia02.jpg"/><br/>
<cfinput type="checkbox" name="ck2" Value=1>Pentagon</td>
<td><img src="../cfdocs/images/artgallery/elecia03.jpg"/><br/>
<cfinput type="checkbox" name="ck3" Value=1>Surfer Dude</td>
<td><img src="../cfdocs/images/artgallery/elecia04.jpg"/><br/>
<cfinput type="checkbox" name="ck4" Value=1>Surfer Girl</td>
</tr>
<tr><td><cfinput type = "Submit" name = "OK" label="OK"></td></tr>
</table>
</cfform>

The first action page zips the files selected from the form, and writes the ZIP file to the hard drive. Also, it includes a form to e-mail the ZIP file:
<!--- Determine the absolute pathname on the server. --->
<cfset thisDir = ExpandPath(".")>

<!--- Create a ZIP file based on the selections from the form. Use the cfzipparam tag to specify the source for each check box selection. --->
<cfzip file="c:\images.zip" source="#thisDir#" action="zip" overwrite="yes">
  <cfif IsDefined("form.ck1")>
    <cfzipparam source="../cfdocs/images/artgallery/elecia01.jpg">
  </cfif>
  <cfif IsDefined("form.ck2")>
    <cfzipparam source="../cfdocs/images/artgallery/elecia02.jpg">
  </cfif>
  <cfif IsDefined("form.ck3")>
    <cfzipparam source="../cfdocs/images/artgallery/elecia03.jpg">
  </cfif>
  <cfif IsDefined("form.ck4")>
    <cfzipparam source="../cfdocs/images/artgallery/elecia04.jpg">
  </cfif>
</cfzip>

<h3>Mail the ZIP File</h3>
<p>Please enter your e-mail address so we can send you the ZIP file as an attachment.</p>
<cfif IsDefined("form.mailto")>
  <cfif form.mailto is not "">
    <cfoutput>
      <cfmail from="coldfusion@adobe.com" to="#form.mailto#"
        subject="see zipped attachment">
        The images you requested are enclosed in a ZIP file.
        <cfmailparam file="#thisDir#/images.zip">
      </cfmail>
    </cfoutput>
  </cfif>
</cfif>
</cfform>

The second action page mails the ZIP file as an attachment:
<h3>Mail the ZIP file</h3>
<p>Your file has been mailed to you.</p>
<cfset eMail="#form.MailTo#">
<cfset zipPath="#form.zipPath#">
<cfmail from="coldfusion@adobe.com" to="#eMail#" subject="see zipped attachment">
    The images you requested are enclosed in a ZIP file.
    <cfmailparam file="#zipPath#">
</cfmail>

To protect a zip archive with a password:

<cfzip action="zip" file="<zip_file_path>" source="<source_dir_path>"
    password="admin" encryptionAlgorithm="AES-128" />
<cfzip action="unzip" file="<zip_file_path>" destination="<dest_dir_path>" >
    <cfzipparam entrypath="sample.txt" password="admin" encryptionAlgorithm="AES-128">
</cfzip>
ColdFusion Functions

The following list shows the ColdFusion Markup Language (CFML) functions.

- New Functions in ColdFusion 10
- Functions by category
- Function changes since ColdFusion 5
- Functions a-b
- Functions c-d
- Functions e-g
- Functions h-im
- Functions in-k
- Functions l
- Functions m-r
- Functions s
- Functions t-z

#back to top
## New Functions in ColdFusion 10

<table>
<thead>
<tr>
<th>ArraySlice</th>
<th>ImageMakeTranslucent</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayEach</td>
<td>Invoke</td>
</tr>
<tr>
<td>ArrayFilter</td>
<td>IsClosure</td>
</tr>
<tr>
<td>ArrayFindAll</td>
<td>ListFilter</td>
</tr>
<tr>
<td>ArrayFindAllNoCase</td>
<td>LSDateTimeFormat</td>
</tr>
<tr>
<td>CacheIdExists</td>
<td>ListRemoveDuplicates</td>
</tr>
<tr>
<td>CacheRegionNew</td>
<td>OnWSAuthenticate</td>
</tr>
<tr>
<td>CacheRegionRemove</td>
<td>ORMIndex</td>
</tr>
<tr>
<td>CacheRemoveAll</td>
<td>ORMIndexPurge</td>
</tr>
<tr>
<td>Canonicalize</td>
<td>ORMSearch</td>
</tr>
<tr>
<td>CacheRegionExists</td>
<td>ORMSearchOffline</td>
</tr>
<tr>
<td>CallStackDump</td>
<td>ReEscape</td>
</tr>
<tr>
<td>CallStackGet</td>
<td>RestInitApplication</td>
</tr>
<tr>
<td>CSRFGenerateToken</td>
<td>RemoveCachedQuery</td>
</tr>
<tr>
<td>CSRFVerifyToken</td>
<td>RestDeleteApplication</td>
</tr>
<tr>
<td>DateTimeFormat</td>
<td>RestSetResponse</td>
</tr>
<tr>
<td>DecodeForHTML</td>
<td>SessionRotate</td>
</tr>
<tr>
<td>DecodeFromURL</td>
<td>SessionGetMetaData</td>
</tr>
<tr>
<td>DirectoryCopy</td>
<td>SessionInvalidate</td>
</tr>
<tr>
<td>EncodeForHTML</td>
<td>StructEach</td>
</tr>
<tr>
<td>EncodeForCSS</td>
<td>StructFilter</td>
</tr>
<tr>
<td>EncodeForHTMLAttribute</td>
<td>WSGetAllChannels</td>
</tr>
<tr>
<td>EncodeForJavaScript</td>
<td>WSGetSubscribers</td>
</tr>
<tr>
<td>EncodeForURL</td>
<td>WSPublish</td>
</tr>
<tr>
<td>EncodeForXML</td>
<td>WSSendMessage</td>
</tr>
<tr>
<td>FileGetMimeType</td>
<td></td>
</tr>
<tr>
<td>GetApplicationMetadata</td>
<td></td>
</tr>
<tr>
<td>GetCPUUsage</td>
<td></td>
</tr>
<tr>
<td>GetTotalSpace</td>
<td></td>
</tr>
<tr>
<td>GetFreeSpace</td>
<td></td>
</tr>
<tr>
<td>GetSystemFreeMemory</td>
<td></td>
</tr>
<tr>
<td>GetSystemTotalMemory</td>
<td></td>
</tr>
<tr>
<td>HMAC</td>
<td></td>
</tr>
<tr>
<td>ImageCreateCaptcha</td>
<td></td>
</tr>
<tr>
<td>ImageMakeColorTransparent</td>
<td></td>
</tr>
</tbody>
</table>

*#back to top*
Functions by category

The following section list functions by their category or purpose.

- Array functions
- Cache functions
- Conversion functions
- Date and time functions
- Data output functions
- Debugging functions
- Decision functions
- Display and formatting functions
- Dynamic evaluation functions
- Exception handling functions
- Extensibility functions
- Flow control functions
- Full-text search functions
- Image functions
- International functions
- List functions
- Mathematical functions
- Microsoft office integration functions
- ORM functions
- Other functions
- Query functions
- Security functions
- Spreadsheet functions
- String functions
- Structure functions
- System functions
- Transaction functions
- XML functions
- Mobile Functions
## Array functions

<table>
<thead>
<tr>
<th>ArrayAppend</th>
<th>ArrayIsDefined</th>
<th>ArrayNew</th>
<th>ArraySum</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayAvg</td>
<td>ArrayIsEmpty</td>
<td>ArrayPrepend</td>
<td>ArraySwap</td>
</tr>
<tr>
<td>ArrayClear</td>
<td>arrayLen</td>
<td>ArrayResize</td>
<td>ArrayToList</td>
</tr>
<tr>
<td>ArrayDeleteAt</td>
<td>ArrayMax</td>
<td>ArraySet</td>
<td>IsArray</td>
</tr>
<tr>
<td>ArrayInsertAt</td>
<td>ArrayMin</td>
<td>ArraySort</td>
<td>ListToArray</td>
</tr>
<tr>
<td>ArrayContains</td>
<td>ArrayDelete</td>
<td>ArrayFind</td>
<td>ArrayFindNoCase</td>
</tr>
<tr>
<td>ArrayEach</td>
<td>ArrayFilter</td>
<td>ArrayFindAll</td>
<td>ArrayFindAllNoCase</td>
</tr>
<tr>
<td>ArraySlice</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Cache functions

<table>
<thead>
<tr>
<th>CacheGet</th>
<th>CacheGetMetadata</th>
<th>CachePut</th>
<th>CacheSetProperties</th>
</tr>
</thead>
<tbody>
<tr>
<td>CacheGetAllIds</td>
<td>CacheGetProperties</td>
<td>CacheRemove</td>
<td>CacheIdExists</td>
</tr>
<tr>
<td>CacheRegionNew</td>
<td>CacheRegionRemove</td>
<td>CacheRemoveAll</td>
<td>Canonicalize</td>
</tr>
</tbody>
</table>
## Conversion functions

<table>
<thead>
<tr>
<th>ArrayToList</th>
<th>DotNetToCFType</th>
<th>ToBinary</th>
<th>ToXml</th>
<th>XmlFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinaryDecode</td>
<td>Hash</td>
<td>ToScript</td>
<td>XmlParse</td>
<td></td>
</tr>
<tr>
<td>BinaryEncode</td>
<td>LCase</td>
<td>ToString</td>
<td>XmlParse</td>
<td></td>
</tr>
<tr>
<td>CharSetDecode</td>
<td>ListToArray</td>
<td>URLDecode</td>
<td>XmlParse</td>
<td></td>
</tr>
<tr>
<td>CharSetEncode</td>
<td>SerializeJSON</td>
<td>URLEncodedFormat</td>
<td>XmlParse</td>
<td></td>
</tr>
<tr>
<td>DeserializerJSON</td>
<td>ToBase64</td>
<td>Val</td>
<td>XmlTransform</td>
<td></td>
</tr>
</tbody>
</table>
**Date and time functions**

<table>
<thead>
<tr>
<th>CreateDate</th>
<th>DateFormat</th>
<th>GetTimeZoneInfo</th>
<th>MonthAsString</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateDateTime</td>
<td>DatePart</td>
<td>Hour</td>
<td>Now</td>
</tr>
<tr>
<td>CreateODBCDate</td>
<td>Day</td>
<td>IsDate</td>
<td>ParseDateTime</td>
</tr>
<tr>
<td>CreateODBCDateTime</td>
<td>DayOfWeek</td>
<td>IsLeapYear</td>
<td>Quarter</td>
</tr>
<tr>
<td>CreateODBCTime</td>
<td>DayOfWeekAsString</td>
<td>IsNumericDate</td>
<td>Second</td>
</tr>
<tr>
<td>CreateTime</td>
<td>DaysOfYear</td>
<td>LSDateFormat</td>
<td>TimeFormat</td>
</tr>
<tr>
<td>CreateTimeSpan</td>
<td>DaysInMonth</td>
<td>LSIsDate</td>
<td>Week</td>
</tr>
<tr>
<td>DateAdd</td>
<td>DaysInYear</td>
<td>LSParseDateTime</td>
<td>Year</td>
</tr>
<tr>
<td>DateCompare</td>
<td>FirstDayOfMonth</td>
<td>LSTimeFormat</td>
<td>DateTimeFormat</td>
</tr>
<tr>
<td>DateConvert</td>
<td>GetHttpTimeString</td>
<td>GetTickCount</td>
<td></td>
</tr>
<tr>
<td>DateDiff</td>
<td>GetTimeZoneInfo</td>
<td>GetTickCount</td>
<td></td>
</tr>
</tbody>
</table>
Data output functions

WriteDump

WriteLog

WriteOutput
### Debugging functions

| Trace |
## Decision functions

<table>
<thead>
<tr>
<th>DirectoryExists</th>
<th>IsDefined</th>
<th>IsPDPFile</th>
<th>IsXmlElem</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileExists</td>
<td>IsInstanceOf</td>
<td>IsPDFObject</td>
<td>IsXmlNode</td>
</tr>
<tr>
<td>FileIsEOF</td>
<td>IsJSON</td>
<td>IsQuery</td>
<td>IsXmlRoot</td>
</tr>
<tr>
<td>IIf</td>
<td>IsK2ServerABroker</td>
<td>IsSimpleValue</td>
<td>LSIsCurrency</td>
</tr>
<tr>
<td>IsArray</td>
<td>IsK2ServerDocCountExceeded</td>
<td>IsStruct</td>
<td>LSIsDate</td>
</tr>
<tr>
<td>IsBinary</td>
<td>IsK2ServerOnline</td>
<td>IsUserInAnyRole</td>
<td>LSIsNumeric</td>
</tr>
<tr>
<td>IsBoolean</td>
<td>IsLeapYear</td>
<td>IsValid</td>
<td>StructIsEmpty</td>
</tr>
<tr>
<td>IsCustomFunction</td>
<td>IsNumeric</td>
<td>IsWDDX</td>
<td>StructIsEmpty</td>
</tr>
<tr>
<td>IsDate</td>
<td>IsNumericDate</td>
<td>IsXML</td>
<td>StructIsEmpty</td>
</tr>
<tr>
<td>IsDebugMode</td>
<td>IsObject</td>
<td>IsXmlAttribute</td>
<td>YesNoFormat</td>
</tr>
<tr>
<td>IsDDX</td>
<td>IsNull</td>
<td>IsXmlDoc</td>
<td></td>
</tr>
</tbody>
</table>
## Display and formatting functions

<table>
<thead>
<tr>
<th>AjaxLink</th>
<th>GetLocaleDisplayname</th>
<th>LSIsDate</th>
<th>ParagraphFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>AjaxOnLoad</td>
<td>GetLocaleDisplayName</td>
<td>LSIsNumber</td>
<td>RJustify</td>
</tr>
<tr>
<td>CJustify</td>
<td>HTMLCodeFormat</td>
<td>LSNumber</td>
<td>StripCR</td>
</tr>
<tr>
<td>DateFormat</td>
<td>HTMLEditFormat</td>
<td>LSDate</td>
<td>TimeFormat</td>
</tr>
<tr>
<td>DecimalFormat</td>
<td>LJustify</td>
<td>LSDateTim</td>
<td>YesNoFormat</td>
</tr>
<tr>
<td>DollarFormat</td>
<td>LSCurrencyFormat</td>
<td>LSParseCurrency</td>
<td></td>
</tr>
<tr>
<td>FormatBaseN</td>
<td>LSDateFormat</td>
<td>LSParseDateTime</td>
<td></td>
</tr>
<tr>
<td>GetLocale</td>
<td>LSEuroCurrencyFormat</td>
<td>LSParseEuroCurrency</td>
<td></td>
</tr>
<tr>
<td>GetLocaleDisplayName</td>
<td>LSNumberFormat</td>
<td>LSParseNumber</td>
<td></td>
</tr>
<tr>
<td>LJustify</td>
<td>LSIsCurrency</td>
<td>LSTimeFormat</td>
<td></td>
</tr>
<tr>
<td>NumberFormat</td>
<td>LSIsCurrency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ParagraphFormat</td>
<td>RJustify</td>
<td></td>
<td></td>
</tr>
<tr>
<td>StripCR</td>
<td>TimeFormat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YesNoFormat</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Dynamic evaluation functions

<table>
<thead>
<tr>
<th>DE</th>
<th>Evaluate</th>
<th>IIf</th>
<th>PrecisionEvaluate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetVariable</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exception handling functions

**Throw**
### Extensibility functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObject</td>
</tr>
<tr>
<td>DotNetToCFType</td>
</tr>
<tr>
<td>GetComponentMetaData</td>
</tr>
<tr>
<td>GetGatewayHelper</td>
</tr>
<tr>
<td>IsInstanceOf</td>
</tr>
<tr>
<td>ReleaseComObject</td>
</tr>
<tr>
<td>SendGatewayMessage</td>
</tr>
<tr>
<td>ToScript</td>
</tr>
</tbody>
</table>
### Flow control functions

<table>
<thead>
<tr>
<th>Throw</th>
<th>Location</th>
</tr>
</thead>
</table>

Full-text search functions

History

ColdFusion MX 6.1: These functions are deprecated. They might not work, and might cause errors, in a future release.

<table>
<thead>
<tr>
<th>GetK2ServerDocCountLimit</th>
<th>IsK2ServerABroker</th>
<th>IsK2ServerOnline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IsK2ServerDocCountExceeded</td>
<td></td>
</tr>
</tbody>
</table>
## Image functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageAddBorderColor</td>
<td>ImageDrawRect</td>
<td>ImageNewImageInfo</td>
<td>ImageSetDrawingColor</td>
</tr>
<tr>
<td>ImageBlur</td>
<td>ImageDrawRoundRect</td>
<td>ImageNegative</td>
<td>ImageSetDrawingStroke</td>
</tr>
<tr>
<td>ImageClearRect</td>
<td>ImageDrawText</td>
<td>ImageNew</td>
<td>ImageSetDrawingTransparency</td>
</tr>
<tr>
<td>ImageCopy</td>
<td>ImageFlip</td>
<td>ImageOverlay</td>
<td>ImageSharpen</td>
</tr>
<tr>
<td>ImageCrop</td>
<td>ImageGetBlob</td>
<td>ImagePaste</td>
<td>ImageShear</td>
</tr>
<tr>
<td>ImageDrawArc</td>
<td>ImageGetBufferedImage</td>
<td>ImageRead</td>
<td>ImageShearDrawingAxis</td>
</tr>
<tr>
<td>ImageDrawBeveledRect</td>
<td>ImageDrawCubicCurve</td>
<td>ImageReadBase64</td>
<td>ImageTranslate</td>
</tr>
<tr>
<td>ImageDrawLine</td>
<td>ImageDrawLine</td>
<td>ImageRotate</td>
<td>ImageTranslateDrawingAxis</td>
</tr>
<tr>
<td>ImageDrawOval</td>
<td>ImageDrawOval</td>
<td>ImageRotateDrawingAxis</td>
<td>ImageWrite</td>
</tr>
<tr>
<td>ImageDrawPoint</td>
<td>ImageDrawQuadraticCurve</td>
<td>ImageScaleToFit</td>
<td>ImageWriteBase64</td>
</tr>
<tr>
<td>ImageDrawRect</td>
<td>ImageGrayscale</td>
<td>ImageSetAntialiasing</td>
<td>ImageXORDrawingMode</td>
</tr>
<tr>
<td>ImageFlip</td>
<td>ImageGetEXIFMetaData</td>
<td>ImageSetBackgroundColor</td>
<td>IsImage</td>
</tr>
<tr>
<td>ImageGetBlob</td>
<td>ImageGetEXIFTag</td>
<td>ImageSetBackgroundColor</td>
<td>IsImageFile</td>
</tr>
<tr>
<td>ImageGetBufferedImage</td>
<td>ImageGetHeight</td>
<td>ImageSetDrawingColor</td>
<td></td>
</tr>
<tr>
<td>ImageDrawArc</td>
<td>ImageGetIPTCTag</td>
<td>ImageSetDrawingStroke</td>
<td></td>
</tr>
<tr>
<td>ImageDrawLine</td>
<td>ImageGetIPTCMetaData</td>
<td>ImageSetDrawingTransparency</td>
<td></td>
</tr>
<tr>
<td>ImageDrawOval</td>
<td>ImageGetWidth</td>
<td>ImageSharpen</td>
<td></td>
</tr>
<tr>
<td>ImageDrawPoint</td>
<td>ImageGrayscale</td>
<td>ImageShear</td>
<td></td>
</tr>
<tr>
<td>ImageDrawQuadraticCurve</td>
<td>ImageGetEXIFMetaData</td>
<td>ImageShearDrawingAxis</td>
<td></td>
</tr>
<tr>
<td>ImageFlip</td>
<td>ImageGetEXIFTag</td>
<td>ImageTranslate</td>
<td></td>
</tr>
<tr>
<td>ImageGetBufferedImage</td>
<td>ImageGetHeight</td>
<td>ImageTranslateDrawingAxis</td>
<td></td>
</tr>
<tr>
<td>ImageGetEXIFMetaData</td>
<td>ImageGetIPTCMetaData</td>
<td>ImageWrite</td>
<td></td>
</tr>
<tr>
<td>ImageGetEXIFTag</td>
<td>ImageGetWidth</td>
<td>ImageWriteBase64</td>
<td></td>
</tr>
<tr>
<td>ImageGetHeight</td>
<td>ImageGetIPTCMetaData</td>
<td>ImageXORDrawingMode</td>
<td></td>
</tr>
<tr>
<td>ImageGetWidth</td>
<td>ImageGetIPTCMetaData</td>
<td>IsImage</td>
<td></td>
</tr>
<tr>
<td>ImageGrayscale</td>
<td>ImageGetEXIFMetaData</td>
<td>IsImageFile</td>
<td></td>
</tr>
</tbody>
</table>
### International functions

<table>
<thead>
<tr>
<th>DateConvert</th>
<th>GetTimeZoneInfo</th>
<th>LSIsDate</th>
<th>LSIsCurrency</th>
<th>LSIsNumeric</th>
<th>LSNumberFormat</th>
<th>LSParseCurrency</th>
<th>LSParseEuroCurrency</th>
<th>LSParseDateTime</th>
<th>LSDateFormat</th>
<th>LSParseNumber</th>
<th>LSTimeFormat</th>
<th>SetLocale</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetEncoding</td>
<td>GetHttpTimeString</td>
<td>GetLocale</td>
<td>GetLocaleDisplayName</td>
<td>GetLocale</td>
<td>GetHttpTimeString</td>
<td>GetLocaleDisplayName</td>
<td>GetLocaleDisplayName</td>
<td>GetLocaleDisplayName</td>
<td>GetLocaleDisplayName</td>
<td>GetLocaleDisplayName</td>
<td>GetLocaleDisplayName</td>
<td>GetLocaleDisplayName</td>
</tr>
</tbody>
</table>
### List functions

<table>
<thead>
<tr>
<th>ArraySort</th>
<th>ArrayToList</th>
<th>Asc</th>
<th>CJustify</th>
<th>Compare</th>
<th>CompareNoCase</th>
<th>Decrypt</th>
<th>Encrypt</th>
<th>Find</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListAppend</td>
<td>ListContains</td>
<td>ListContainsNoCase</td>
<td>ListDeleteAt</td>
<td>ListFind</td>
<td>ListFindNoCase</td>
<td>ListFirst</td>
<td>ListGetAt</td>
<td>ListInsertAt</td>
</tr>
<tr>
<td>ListQualify</td>
<td>ListReduce</td>
<td>ListRest</td>
<td>ListRemoveDuplicate</td>
<td>ListSetAt</td>
<td>ListSort</td>
<td>ListToArray</td>
<td>ListValueCount</td>
<td>ListValueCountNoCase</td>
</tr>
</tbody>
</table>
## Mathematical functions

<table>
<thead>
<tr>
<th>Aabs</th>
<th>ACos</th>
<th>ArrayAvg</th>
<th>ArraySum</th>
<th>ASin</th>
<th>Atn</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitNot</td>
<td>BitOr</td>
<td>BitSHLN</td>
<td>BitSHRN</td>
<td>BitXor</td>
<td>Ceiling</td>
</tr>
<tr>
<td>Cos</td>
<td>DecrementValue</td>
<td>Exp</td>
<td>Fix</td>
<td>FormatBaseN</td>
<td>IncrementValue</td>
</tr>
<tr>
<td>InputBaseN</td>
<td>Int</td>
<td>Log</td>
<td>Log10</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>Pi</td>
<td>PrecisionEvaluate</td>
<td>Rand</td>
<td>Randomize</td>
<td>RandRange</td>
<td>Round</td>
</tr>
<tr>
<td>Sgn</td>
<td>Sin</td>
<td>Sqr</td>
<td>Tan</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Microsoft office integration functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsSpreadsheetFileType</code></td>
<td><code>SpreadsheetDeleteColumns</code></td>
<td><code>SpreadsheetRead</code></td>
<td><code>SpreadsheetSetActiveSheet</code></td>
</tr>
<tr>
<td><code>IsSpreadsheetObject</code></td>
<td><code>SpreadsheetDeleteRows</code></td>
<td><code>SpreadsheetRemoveSheet</code></td>
<td><code>SpreadsheetSetFooter</code></td>
</tr>
<tr>
<td><code>SpreadsheetAddColumn</code></td>
<td><code>SpreadsheetDeleteRow</code></td>
<td><code>SpreadsheetSetInfo</code></td>
<td><code>SpreadsheetSetHeader</code></td>
</tr>
<tr>
<td><code>SpreadsheetAddFreezePane</code></td>
<td><code>SpreadsheetDeleteRows</code></td>
<td><code>SpreadsheetMergeCells</code></td>
<td><code>SpreadsheetSetColumnWidth</code></td>
</tr>
<tr>
<td><code>SpreadsheetAddImage</code></td>
<td><code>SpreadsheetFormatCell</code></td>
<td><code>SpreadsheetNew</code></td>
<td><code>SpreadsheetShiftColumns</code></td>
</tr>
<tr>
<td><code>SpreadsheetAddInfo</code></td>
<td><code>SpreadsheetFormatCellRange</code></td>
<td><code>SpreadsheetSetActiveSheetNumber</code></td>
<td><code>SpreadsheetShiftRows</code></td>
</tr>
<tr>
<td><code>SpreadsheetAddRows</code></td>
<td><code>SpreadsheetFormatColumn</code></td>
<td><code>SpreadsheetSetActiveSheet</code></td>
<td><code>SpreadsheetSetRowHeight</code></td>
</tr>
<tr>
<td><code>SpreadsheetAddSplitPane</code></td>
<td><code>SpreadsheetFormatColumns</code></td>
<td><code>SpreadsheetSetCellComment</code></td>
<td><code>SpreadsheetWriteTrace</code></td>
</tr>
<tr>
<td><code>SpreadsheetCreateSheet</code></td>
<td><code>SpreadsheetFormatRow</code></td>
<td><code>SpreadsheetSetCellFormula</code></td>
<td></td>
</tr>
<tr>
<td><code>SpreadsheetDeleteColumn</code></td>
<td><code>SpreadsheetFormatRows</code></td>
<td><code>SpreadsheetSetCellValue</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>SpreadsheetGetCellComment</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>SpreadsheetGetCellFormula</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>SpreadsheetGetCellValue</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## ORM functions

<table>
<thead>
<tr>
<th>EntityDelete</th>
<th>EntityLoad</th>
<th>EntityLoadByExample</th>
<th>EntityLoadByPK</th>
<th>EntityMerge</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityNew</td>
<td>EntityReload</td>
<td>EntitySave</td>
<td>EntitytoQuery</td>
<td>ORMClearSession</td>
</tr>
<tr>
<td>ORMCloseSession</td>
<td>ORMEvictCollection</td>
<td>ORMEvictEntity</td>
<td>ORMEvictQueries</td>
<td>ORMExecuteQuery</td>
</tr>
<tr>
<td>ORMFlush</td>
<td>ORMGetSession</td>
<td>ORMGetSessionFactor</td>
<td>ORMReload</td>
<td></td>
</tr>
</tbody>
</table>
Other functions

<table>
<thead>
<tr>
<th>ApplicationStop</th>
<th>GetBaseTemplatePath</th>
<th>IsLocalHost</th>
<th>URLSessionFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateUUID</td>
<td>GetClientVariablesList</td>
<td>ObjectEquals</td>
<td>WriteDump</td>
</tr>
<tr>
<td>DeleteClientVariables</td>
<td>GetLocalHostIP</td>
<td>ObjectLoad</td>
<td>WriteLog</td>
</tr>
<tr>
<td>GetBaseTagData</td>
<td>GetBaseTagList</td>
<td>ObjectSave</td>
<td>WriteOutput</td>
</tr>
<tr>
<td>GetLocalHostIP</td>
<td></td>
<td>PreserveSingleQuotes</td>
<td></td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
### Query functions

<table>
<thead>
<tr>
<th>IsQuery</th>
<th>QueryAddRow</th>
<th>QueryNew</th>
<th>QuotedValueList</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryAddColumn</td>
<td>QueryConvertForGrid</td>
<td>QuerySetCell</td>
<td>ValueList</td>
</tr>
</tbody>
</table>
Security functions

- Decrypt
- Decrypt
- DecryptBinary
- Encrypt
- Encrypt
- EncryptBinary
- GetAuthUser
- GenerateSecretKey
- GetTempDirectory
- GetTempFile
- GetUserRoles
- Hash
- IsUserInAnyRole
- IsUserInRole
- IsUserLoggedIn
- VerifyClient
- VerifyClient
- VerifyClient
- GeneratePBKDFKey
## Spreadsheet functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpreadsheetAddColumn</td>
<td>SpreadsheetColumnWidth</td>
<td>SpreadsheetSetCellComment</td>
<td>SpreadsheetSetColumnWidth</td>
</tr>
<tr>
<td>SpreadsheetAddImage</td>
<td></td>
<td>SpreadsheetSetCellValue</td>
<td>SpreadsheetSetFooter</td>
</tr>
<tr>
<td>SpreadsheetFreezePane</td>
<td></td>
<td>SpreadsheetSetFormula</td>
<td>SpreadsheetSetHeader</td>
</tr>
<tr>
<td>SpreadsheetAddInfo</td>
<td></td>
<td>SpreadsheetSetRange</td>
<td>SpreadsheetSetRowHeight</td>
</tr>
<tr>
<td>SpreadsheetAddRow</td>
<td></td>
<td>SpreadsheetSetRow</td>
<td>SpreadsheetShiftColumns</td>
</tr>
<tr>
<td>SpreadsheetAddRows</td>
<td></td>
<td>SpreadsheetShiftRows</td>
<td>SpreadsheetShiftRows</td>
</tr>
<tr>
<td>SpreadsheetAddSplitPane</td>
<td></td>
<td>SpreadsheetWrite</td>
<td>SpreadsheetWrite</td>
</tr>
<tr>
<td>SpreadsheetCreateSheet</td>
<td></td>
<td>SpreadsheetWriteBinary</td>
<td>SpreadsheetAddPagebreaks</td>
</tr>
<tr>
<td>SpreadsheetDeleteColumn</td>
<td></td>
<td>SpreadsheetWriteCell</td>
<td></td>
</tr>
<tr>
<td>SpreadsheetDeleteRows</td>
<td></td>
<td>SpreadsheetWriteCellRange</td>
<td></td>
</tr>
<tr>
<td>SpreadsheetFormatCell</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetFormatColumn</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetFormatCellRange</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetFormatColumns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetFormatRow</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetFormatRows</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetGetCellValue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetGetCellComment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetGetCellFormula</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetGetColumnWidth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetGetFooter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetGetHeader</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetGetRowHeight</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetMergeCells</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetNew</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetRead</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetReadBinary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetRemoveSheet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetSetActiveSheet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetSetActiveSheetNumber</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetSetCellComment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetSetCellValue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpreadsheetSetCellFormula</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
String functions

History

ColdFusion MX: ColdFusion now supports the Java UCS-2 representation of Unicode character values 0-65535. (Earlier releases supported ASCII values.) String-processing functions process any of these characters (including ASCII 0 (NUL) characters), and continue counting subsequent characters of the string, if any. (In earlier releases, some string-processing functions processed the ASCII 0 (NUL) character, but did not process subsequent characters of the string.)

<table>
<thead>
<tr>
<th>Asc</th>
<th>Hash</th>
<th>LSParseDateTime</th>
<th>ReplaceList</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinaryDecode</td>
<td>HTMLCodeFormat</td>
<td>LSParseEuroCurrency</td>
<td>Reverse</td>
</tr>
<tr>
<td>BinaryEncode</td>
<td>HTMLEditFormat</td>
<td>Mid</td>
<td>Right</td>
</tr>
<tr>
<td>CharsetDecode</td>
<td>Insert</td>
<td>MonthAsString</td>
<td>RJustify</td>
</tr>
<tr>
<td>CharsetEncode</td>
<td>JSStringFormat</td>
<td>ParagraphFormat</td>
<td>SpanIncluding</td>
</tr>
<tr>
<td>Chr</td>
<td>LCase</td>
<td>ParseDateTime</td>
<td>StripCR</td>
</tr>
<tr>
<td>CJustify</td>
<td>Left</td>
<td>REFind</td>
<td>ToBase64</td>
</tr>
<tr>
<td>Compare</td>
<td>Len</td>
<td>REFindNoCase</td>
<td>ToBinary</td>
</tr>
<tr>
<td>CompareNoCase</td>
<td>LJustify</td>
<td>REMatch</td>
<td>ToString</td>
</tr>
<tr>
<td>DayOfWeekAsString</td>
<td>ListValueCount</td>
<td>REMatchNoCase</td>
<td>Trim</td>
</tr>
<tr>
<td>Decrypt</td>
<td>LSParseNumber</td>
<td>RemoveChars</td>
<td>UCase</td>
</tr>
<tr>
<td>Encrypt</td>
<td>LTrim</td>
<td>RepeatString</td>
<td>URLDecode</td>
</tr>
<tr>
<td>Find</td>
<td>ListValueCountNoCase</td>
<td>Reverse</td>
<td>URLEncodedFormat</td>
</tr>
<tr>
<td>FindNoCase</td>
<td>LSIsDate</td>
<td>Right</td>
<td>Val</td>
</tr>
<tr>
<td>FindOneOf</td>
<td>LSIsNumeric</td>
<td>REReplace</td>
<td>Wrap</td>
</tr>
<tr>
<td>FormatBaseN</td>
<td>LSParseCurrency</td>
<td>REReplaceNoCase</td>
<td>XmlFormat</td>
</tr>
<tr>
<td>GenerateSecretKey</td>
<td>LSIsCurrency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GetToken</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See also Conversion functions.
## Structure functions

<table>
<thead>
<tr>
<th>Duplicate</th>
<th>StructCount</th>
<th>StructGet</th>
<th>StructKeyList</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsStruct</td>
<td>StructDelete</td>
<td>StructInsert</td>
<td>StructNew</td>
</tr>
<tr>
<td>StructAppend</td>
<td>StructFind</td>
<td>StructIsEmpty</td>
<td>StructSort</td>
</tr>
<tr>
<td>StructClear</td>
<td>StructFindKey</td>
<td>StructKeyArray</td>
<td>StructUpdate</td>
</tr>
<tr>
<td>StructCopy</td>
<td>StructFindValue</td>
<td>StructKeyExists</td>
<td></td>
</tr>
</tbody>
</table>
# System functions

<table>
<thead>
<tr>
<th>DirectoryExists</th>
<th>FileSetAttribute</th>
<th>GetFunctionList</th>
<th>GetTempFile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicate</td>
<td>FileSetLastModified</td>
<td>GetHttpRequestData</td>
<td>GetTemplatePath</td>
</tr>
<tr>
<td>ExpandPath</td>
<td>DirectoryDelete</td>
<td>GetLocale</td>
<td>GetTickCount</td>
</tr>
<tr>
<td>FileClose</td>
<td>FileSeek</td>
<td>GetLocaleDisplayName</td>
<td>GetWriteableImageFormats</td>
</tr>
<tr>
<td>FileCopy</td>
<td>FileWrite</td>
<td>GetMetaData</td>
<td>SetLocale</td>
</tr>
<tr>
<td>FileDelete</td>
<td>GetBaseTemplatePath</td>
<td>GetMetricData</td>
<td>GetProfileString</td>
</tr>
<tr>
<td>FileExists</td>
<td>GetContextRoot</td>
<td>DirectoryList</td>
<td>Sleep</td>
</tr>
<tr>
<td>FileIsEOF</td>
<td>GetCurrentTemplateName</td>
<td>FileSkipBytes</td>
<td>WriteOutput</td>
</tr>
<tr>
<td>FileMove</td>
<td>GetDirectoryFromPath</td>
<td>GetPageContext</td>
<td>DirectoryExists</td>
</tr>
<tr>
<td>FileOpen</td>
<td>GetDirectoryFromPath</td>
<td>GetPrinterInfo</td>
<td>DirectoryCreate</td>
</tr>
<tr>
<td>FileRead</td>
<td>GetEncoding</td>
<td>GetProfileSections</td>
<td>DirectoryRename</td>
</tr>
<tr>
<td>FileReadBinary</td>
<td>GetException</td>
<td>GetProfileString</td>
<td>GetFunctionCalledName</td>
</tr>
<tr>
<td>FileReadLine</td>
<td>GetFileInfo</td>
<td>GetReadableImageFormats</td>
<td></td>
</tr>
<tr>
<td>FileSetAccessMode</td>
<td></td>
<td>GetTempDirectory</td>
<td></td>
</tr>
</tbody>
</table>
## Transaction functions

| TransactionCommit | TransactionRollback | TransactionSetSavePoint |
# XML functions

<table>
<thead>
<tr>
<th>AddSOAPRequestHeader</th>
<th>AddSOAPResponseHeader</th>
<th>GetSOAPRequest</th>
<th>GetSOAPRequestHeader</th>
<th>GetSOAPResponse</th>
<th>GetSOAPResponseHeader</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsSOAPRequest</td>
<td>IsXML</td>
<td>IsXmlAttribute</td>
<td>IsXmlDoc</td>
<td>IsXmlElem</td>
<td>IsXmlNode</td>
</tr>
<tr>
<td>IsXmlRoot</td>
<td>IsWDDX</td>
<td></td>
<td></td>
<td>IsXmlNode</td>
<td></td>
</tr>
<tr>
<td>ToString</td>
<td>XmlChildPos</td>
<td></td>
<td></td>
<td>XmlElemNew</td>
<td></td>
</tr>
<tr>
<td>XmlFormat</td>
<td>XmlGetNodeType</td>
<td></td>
<td></td>
<td>XmlFormat</td>
<td></td>
</tr>
<tr>
<td>XmlNew</td>
<td>XmlParse</td>
<td></td>
<td></td>
<td>XmlValidate</td>
<td></td>
</tr>
<tr>
<td>XmlSearch</td>
<td>XmlTransform</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[back to top](#)
Mobile Functions

See ColdFusion Mobile Functions.
Function changes since ColdFusion 5

The tables in the sub-section lists functions, parameters and values that have changed since ColdFusion 5 and indicate the specific release in which the change was made.

- New functions, parameters, and values
- Deprecated functions, parameters, and values
- Obsolete functions, parameters, and values
## New functions, parameters, and values

New and changed functions are also documented elsewhere in the manual, for the following releases:

- [New Functions in ColdFusion 9 and 9.0.1](#) (in the ColdFusion 9 version of this manual)
- [New Functions in ColdFusion 10](#)
- [New in ColdFusion 11](#)

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter or value</th>
<th>Added in this ColdFusion release</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AjaxLink</strong></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><strong>AjaxOnLoad</strong></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><strong>ArrayIsDefined</strong></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><strong>ArrayAppend</strong></td>
<td><em>merge</em> attribute</td>
<td>ColdFusion 10</td>
</tr>
<tr>
<td><strong>BinaryDecode</strong></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><strong>BinaryEncode</strong></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><strong>cacheGetSession</strong></td>
<td>All</td>
<td>ColdFusion 9.0.1</td>
</tr>
<tr>
<td><strong>CharsetDecode</strong></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><strong>CharsetEncode</strong></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><strong>CreateObject</strong></td>
<td><em>value of the type parameter and associated assembly, server, port, protocol, and secure parameters. WSDL2Java and argStruct parameters for web service objects</em></td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>** portName** parameter</td>
<td></td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><strong>DateAdd</strong></td>
<td><em>key of datepart</em> parameter</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td><strong>DatePart</strong></td>
<td><em>key of datepart</em> parameter</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td><strong>Decrypt</strong></td>
<td><em>IVorSalt and iterations parameters</em></td>
<td>ColdFusion MX 7.0.1</td>
</tr>
<tr>
<td>Function</td>
<td>Parameters</td>
<td>ColdFusion Version</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>DecryptBinary</td>
<td>algorithm and encoding parameters</td>
<td>ColdFusion MX 7.0.1</td>
</tr>
<tr>
<td>DeserializeJSON</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>DotNetToCFType</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Encrypt</td>
<td>IVOrSalt and iterations parameters</td>
<td>ColdFusion MX 7.0.1</td>
</tr>
<tr>
<td>EncryptBinary</td>
<td>All</td>
<td>ColdFusion MX 7.0.1</td>
</tr>
<tr>
<td>FileClose</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileCopy</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileDelete</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileIsEOF</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileMove</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileOpen</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileRead</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileReadBinary</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileReadLine</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileSetAccessMode</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileSetAttribute</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileSetLastModified</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>FileUpload</td>
<td>All</td>
<td>ColdFusion 9.0.1</td>
</tr>
<tr>
<td>FileUploadAll</td>
<td>All</td>
<td>ColdFusion 9.0.1</td>
</tr>
<tr>
<td>FileWrite</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Function</td>
<td>Scope</td>
<td>Version</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>GenerateSecretKey</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>GetGatewayHelper</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>GetAuthUser</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>GetComponentMetaData</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>GetContextRoot</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>GetEncoding</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>GetFileInfo</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>GetLocaleDisplayName</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>GetLocalHostIP</td>
<td>All</td>
<td>ColdFusion MX 7.0.1</td>
</tr>
<tr>
<td>GetMetaData</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>GetPageContext</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>GetPrinterInfo</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>GetProfileSections</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>GetReadableImageFormats</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>GetSOAPRequest</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>GetSOAPRequestHeader</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>GetSOAPResponse</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>GetSOAPResponseHeader</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>GetUserRoles</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>GetWriteableImageFormats</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Hash</td>
<td></td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>ImageAddBorder</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageBlur</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Function</td>
<td>Scope</td>
<td>ColdFusion Version</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------</td>
<td>--------------------</td>
</tr>
<tr>
<td>ImageClearRect</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageCopy</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageCrop</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawArc</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawBeveledRect</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawCubicCurve</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawPoint</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawLine</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawLines</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawOval</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawQuadraticCurve</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawRect</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawRoundRect</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageDrawText</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageFlip</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageGetBlob</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageGetBufferedImage</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageGetEXIFMetadata</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageGetEXIFTag</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageGetHeight</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageGetIPTCMetadata</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageGetIPTCTag</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageGetWidth</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>ImageGrayscale</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>Function</td>
<td>Availability</td>
<td>Release</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>ImageInfo</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageNegative</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageNew</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageOverlay</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImagePaste</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageRead</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageReadBase64</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageResize</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageRotate</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageRotateDrawingAxis</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageScaleToFit</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageSetAntialiasing</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageSetBackgroundColor</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageSetDrawingColor</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageSetDrawingStroke</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageSetDrawingTransparency</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageSharpen</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageShear</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageShearDrawingAxis</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageTranslate</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageTranslateDrawingAxis</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageWrite</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>ImageWriteBase64</td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Function</td>
<td>Scope</td>
<td>Version</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td><code>ImageXORDrawingMode</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsDDX</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsImage</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsImageFile</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsInstanceOf</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsJSON</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsLocalHost</code></td>
<td>All</td>
<td>ColdFusion MX 7.0.1</td>
</tr>
<tr>
<td><code>IsObject</code></td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>IsPDFFile</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsPDFObject</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsSOAPRequest</code></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><code>IsUserInAnyRole</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsUserInRole</code></td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>IsUserLoggedIn</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td><code>IsValid</code></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><code>IsXML</code></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><code>IsXmlAttribute</code></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><code>IsXmlNode</code></td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><code>IsXmlRoot</code></td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td><code>LSTimeFormat</code></td>
<td>I key of mask parameter</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td><code>QueryAddColumn</code></td>
<td>datatype parameter</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td><code>QueryConvertForGrid</code></td>
<td>All</td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Function</td>
<td>Parameter(s)</td>
<td>Version(s)</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>QueryNew</td>
<td>columnTypelist</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>PrecisionEvaluate</td>
<td></td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Rand</td>
<td>algorithm</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>Randomize</td>
<td>algorithm</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>RandRange</td>
<td>algorithm</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>ReleaseComObject</td>
<td></td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td>REMatch</td>
<td></td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>REMatchNoCase</td>
<td></td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>SerializeJSON</td>
<td></td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>SendGatewayMessage</td>
<td></td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>Sleep</td>
<td></td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>SpreadsheetFormatCellRange</td>
<td></td>
<td>ColdFusion 9.0.1</td>
</tr>
<tr>
<td>SpreadsheetRemoveSheet</td>
<td></td>
<td>ColdFusion 9.0.1</td>
</tr>
<tr>
<td>TimeFormat</td>
<td>key of mask</td>
<td>ColdFusion 6.1</td>
</tr>
<tr>
<td>ToScript</td>
<td></td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>URLDecode</td>
<td>charset</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>URLEncodedFormat</td>
<td>charset</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>URLSessionFormat</td>
<td></td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>VerifyClient</td>
<td></td>
<td>ColdFusion 8</td>
</tr>
<tr>
<td>Wrap</td>
<td></td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td>XmlChildPos</td>
<td></td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>XmlElemNew</td>
<td></td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>XmlElemNew</td>
<td>namespace</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>Function</td>
<td>Version</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>XmlGetNodeType</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>XmlNew</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>XmlParse</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>XmlParse validator</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>XmlSearch</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>XmlTransform</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>XmlTransform parameters</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
<tr>
<td>XmlValidate</td>
<td>All</td>
<td>ColdFusion MX 7</td>
</tr>
</tbody>
</table>
# Deprecated functions, parameters, and values

The following functions, parameters, and values are deprecated. Do not use them in ColdFusion applications. They might not work, and might cause an error, in releases later than ColdFusion MX.

> **Updated information for ColdFusion 11**

Please note that there is a new document for ColdFusion 11, located outside of this manual, with additional and more details about deprecated and removed/obsoleted features. See [Deprecated Features](#).  

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter or value</th>
<th>Deprecated as of this ColdFusion release</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetMetricData</td>
<td><code>cachepops</code> parameter</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>GetK2ServerDocCount</td>
<td>All</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td>GetK2ServerDocCountLimit</td>
<td>All</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td>GetTemplatePath</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>IsK2ServerABroker</td>
<td>All</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td>IsK2ServerDocCountExceeded</td>
<td>All</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td>IsK2ServerOnLine</td>
<td>All</td>
<td>ColdFusion MX 6.1</td>
</tr>
<tr>
<td>ParameterExists</td>
<td>All</td>
<td>ColdFusion MX. Use the <code>IsDefined</code> function.</td>
</tr>
<tr>
<td>SetLocale</td>
<td><code>locale = &quot;Spanish (Mexican)&quot;</code> value</td>
<td>ColdFusion MX. Use <code>Spanish (Standard).</code></td>
</tr>
</tbody>
</table>
# Obsolete functions, parameters, and values

The following functions, parameters, and values are obsolete. Do not use them in ColdFusion applications. They do not work in releases later than ColdFusion 5.

> Updated information for ColdFusion 11

Please note that there is a new document for ColdFusion 11, located outside of this manual, with additional and more details about deprecated and removed/obsoleted features. See [Deprecated Features](#).

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter or value</th>
<th>Obsolete as of this ColdFusion release</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthenticatedContext</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>AuthenticatedUser</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>isAuthenticated</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>isAuthorized</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>isProtected</td>
<td>All</td>
<td>ColdFusion MX</td>
</tr>
</tbody>
</table>

[back to top](#)
Functions a-b

- Abs
- ACos
- AddSOAPRequestHeader
- AddSOAPResponseHeader
- AjaxLink
- AjaxOnLoad
- ApplicationStop
- ArrayAppend
- ArrayAvg
- ArrayClear
- ArrayContains
- ArrayDelete
- ArrayDeleteAt
- ArrayEach
- ArrayFilter
- ArrayFind
- ArrayFindAll
- ArrayFindAllNoCase
- ArrayFindNoCase
- ArrayInsertAt
- ArrayIsEmpty
- arrayLen
- ArrayMap
- ArrayMax
- ArrayMin
- ArrayNew
- ArrayPrepend
- ArrayReduce
- ArrayResize
- ArraySet
- ArraySlice
- ArraySort
- ArraySum
- ArraySwap
- ArrayToList
- Asc
- ASin
- Atn
- AuthenticatedContext
- AuthenticatedUser
- BinaryDecode
- BinaryEncode
- BitAnd
- BitMaskClear
- BitMaskRead
- BitMaskSet
- BitNot
- BitOr
- BitSHLN
- BitSHRN
- BitXor
- ArrayDeleteNoCase
Abs

Description

Absolute-value function. The absolute value of a number is the number without its sign.

Returns

The absolute value of a number.

Category

Mathematical functions

Function syntax

\[ \text{Abs}(\text{number}) \]

See also

Sgn

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A number</td>
</tr>
</tbody>
</table>

Example

```html
<h3>Abs Example</h3>
<p>The absolute value of the following numbers: 1, 3, -4, -3.2, 6 is</p>
<cfoutput>
#Abs(1)#,#Abs(3)#,#Abs(-4)#,#Abs(-3.2)#,#Abs(6)#
</cfoutput>

<p>The absolute value of a number is the number without its sign.</p>
```
ACos

Description

Arccosine function. The arccosine is the angle whose cosine is number.

Returns

The arccosine, in radians, of a number.

Category

Mathematical functions

Function syntax

```
ACos(number)
```

See also

Cos, Sin, ASin, Tan, Atn, Pi

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Cosine of an angle. The value must be between -1.0 and 1.0, inclusive.</td>
</tr>
</tbody>
</table>

Usage

The range of the result is 0 to p. To convert degrees to radians, multiply degrees by p/180. To convert radians to degrees, multiply radians by 180/p.

Example
<h3>ACos Example</h3>
<!--- Output the arccosine value. --->
<cfif IsDefined("FORM.CosNum")>
  <cfif IsNumeric(FORM.CosNum)>
    <cfif Abs(FORM.CosNum) LESS THAN OR EQUAL TO 1>
      <cfoutput>ACos(#FORM.CosNum#) = #ACos(FORM.cosNum)# Radians</cfoutput>
    </cfif>
    or<br>
    <cfoutput>ACos(#FORM.CosNum#) = #ACos(FORM.cosNum) * 180/PI()#</cfoutput>
  </cfif>
  <cfelse>
    <!--- If it is empty, output an error message. --->
    <h4>Enter a number between -1 and 1</h4>
  </cfif>
</cfif>
</form>
AddSOAPRequestHeader

Description

Adds a SOAP header to a web service request before making the request.

Returns

Nothing.

Category

XML functions

Function syntax

```
AddSOAPRequestHeader(webservice, namespace, name, value [, mustunderstand])
```

See also

AddSOAPResponseHeader, GetSOAPRequest, GetSOAPRequestHeader, GetSOAPResponse, GetSOAPResponseHeader, IsSOAPRequest; Basic web service concepts in the Developing ColdFusion Applications

History

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>webservice</td>
<td>A web service object as returned from the <code>cfobject</code> tag or the CreateObject function.</td>
</tr>
<tr>
<td>namespace</td>
<td>A string that is the namespace for the header.</td>
</tr>
<tr>
<td>name</td>
<td>A string that contains the name of the SOAP header in the request.</td>
</tr>
<tr>
<td>value</td>
<td>The value for the SOAP header; this can be a CFML XML value.</td>
</tr>
<tr>
<td>mustunderstand</td>
<td>Optional. True or False (default). Sets the SOAP mustunderstand value for this header.</td>
</tr>
</tbody>
</table>

Usage

Used within CFML code by a consumer of a web service before it calls the web service. If you pass XML in the `value` parameter, ColdFusion ignores the namespace and name parameters. If you require a namespace, define it within the XML itself.

Example
There are two parts to this example. The first part is the web service CFC that this function (as well as the other ColdFusion SOAP functions) uses to demonstrate its interaction with a web service. To implement the web service for this function, see the example for AddSOAPResponseHeader. Execute the following example as a client to see how the AddSOAPRequestHeader function operates.

```coldfusion
<!--- Note that you might need to modify the URL in the CreateObject function to match your server and the location of the headerservice.cfc file if it is different than shown here. Likewise for the cfinvoke tag at the end. --->

<h3>AddSOAPRequestHeader Example</h3>
<cfscript>
// Create the web service object.
ws = CreateObject("webservice",
"http://localhost/soapheaders/headerservice.cfc?WSDL");

// Set the username header as a string.
addSOAPRequestHeader(ws, "http://mynamespace/", "username", "tom", false);

// Set the password header as a CFML XML object.
doc = XmlNew();
doc.password = XmlElemNew(doc, "http://mynamespace/", "password");
doc.password.XmlText = "My Voice is my Password";
doc.password.XmlAttributes["xsi:type"] = "xsd:string";
addSOAPRequestHeader(ws, "ignoredNameSpace", "ignoredName", doc);

// Invoke the web service operation.
ret = ws.echo_me("argument");

// Get the first header as an object (string) and as XML.
header = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader");
XMLheader = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader", true);

// Get the second header as an object (string) and as XML.
header2 = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader2");
XMLheader2 = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader2", true);
</cfscript>
<hr>

<cfoutput>
Soap Header value: #HTMLCodeFormat(header)#<br>
Soap Header XML value: #HTMLCodeFormat(XMLheader)#<br>
Soap Header 2 value: #HTMLCodeFormat(header2)#<br>
Soap Header 2 XML value: #HTMLCodeFormat(XMLheader2)#<br>
Return value: #HTMLCodeFormat(ret)#<br>
</cfoutput>
<hr>
</cfoutput>
</cfoutput>

<cfinvoke component="soapheaders.headerservice" method="echo_me"
returnvariable="ret" in_here="hi">
</cfinvoke>
<cfoutput>The cfinvoke tag returned: #ret#</cfoutput>
AddSOAPResponseHeader

**Description**

Adds a SOAP response header to a web service response. Call only from within a CFC web service function that is processing a request as a SOAP web service.

**Returns**

Nothing

**Category**

XML functions

**Function syntax**

```
AddSOAPResponseHeader(namespace, name, value [, mustunderstand])
```

**See also**

AddSOAPRequestHeader, GetSOAPRequest, GetSOAPRequestHeader, GetSOAPResponse, GetSOAPResponseHeader, IsSOAPRequest; Basic web service concepts in the Developing ColdFusion Applications

**History**

ColdFusion MX 7: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>A string that is the namespace for the header.</td>
</tr>
<tr>
<td>name</td>
<td>A string that contains the name of the SOAP header in the request.</td>
</tr>
<tr>
<td>value</td>
<td>The value for the SOAP header; this can be a CFML XML value.</td>
</tr>
<tr>
<td>mustunderstand</td>
<td>Optional. True or False (default). Sets the SOAP mustunderstand value for this header.</td>
</tr>
</tbody>
</table>

**Usage**

Call this function only from within a CFC web service function. It throws an error if it is invoked in a context that is not a web service request. If you pass XML in the value parameter, ColdFusion ignores the namespace and name parameters. If you require a namespace, define it within the XML itself. Use the IsSOAPRequest function to determine if the CFC is running as a web service.

**Example**
This example creates a CFC web service that illustrates the operation of the `AddSOAPResponseHeader` function and also provides a web service that illustrates the operation of other ColdFusion SOAP functions. Save the following code as `headerservice.cfc` in a folder called `soapheaders` under your web root. Test its operation, and specifically the operation of the `AddSOAPResponseHeader` function, by executing the examples that invoke this web service. For example, see the example for `AddSOAPRequestHeader`.

```cfml
<h3>AddSOAPResponseHeader Example</h3>
<!--- The headerservice.cfc CFC Web Service.--->
<cfcomponent displayName="tester" hint="Test for SOAP headers">
  <cffunction name="echo_me" access="remote" output="false" returntype="string" displayname="Echo Test" hint="Header test">
    <cfargument name="in_here" required="true" type="string">
    <cfset isSOAP = isSOAPRequest()>
    <cfif isSOAP>
      <!--- Get the first header as a string and as XML. --->
      <cfset username = getSOAPRequestHeader("http://mynamespace/", "username")>
      <cfset return = "The service saw username: " & username>
      <cfset xmlusername = getSOAPRequestHeader("http://mynamespace/", "username", "TRUE")>
      <cfset return = return & "<br> as XML: " & xmlusername>
      <!--- Get the second header as a string and as XML. --->
      <cfset password = getSOAPRequestHeader("http://mynamespace/", "password")>
      <cfset return = return & "The service saw password: " & password>
      <cfset xmlpassword = getSOAPRequestHeader("http://mynamespace/", "password", "TRUE")>
      <cfset return = return & "<br> as XML: " & xmlpassword>
      <!--- Add a header as a string. --->
      <cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED VALUE", false)>
      <!--- Add a second header using a CFML XML value. --->
      <cfset doc = XmlNew()>
      <cfset x = XmlElemNew(doc, "http://www.tomj.org/myns", "returnheader2")>
      <cfset x.XmlText = "hey man, here I am in XML">
      <cfset x.XmlAttributes["xsi:type"] = "xsd:string">
      <cfset tmp = addSOAPResponseHeader("ignoredNameSpace", "ignoredName", x)>
      <cfelse>
        <!--- Add a header as a string - Must generate error! --->
        <cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED VALUE", false)>
        <!--- Not invoked as a web service" --->
        <cfset return = "Not invoked as a web service">
      </cfif>
    </cfif>
  </cffunction>
</cfcomponent>
```
**AjaxLink**

**Description**

Causes an HTML `href` attribute to display link results in the current Ajax container. When the browser follows a link that is specified by this function, the HTTP response does not replace the current page; instead, it populates the containing `cfdiv`, `cflayoutarea`, `cfpod`, or `cfwindow` control.

**Returns**

Code that causes the linked page to be displayed in the containing control.

**Category**

Display and formatting functions

**Function syntax**

```
AjaxLink(URL)
```

**See also**

cfdiv, cflayoutarea, cfpod, cfwindow, Using Ajax User Interface Components and Features in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>The URL of the link.</td>
</tr>
</tbody>
</table>

**Usage**

This function has an effect only when it is used to specify the URL of an `href` attribute when the HTML `a` tag is inside a `cfdiv`, `cflayoutarea`, `cfpod`, or `cfwindow` control. Otherwise, the link has its normal effect. To prevent cross-site scripting, ColdFusion does not load a remote web page.

**Example**

```
<cfpod height="600" width="600" name="podTest">
  <a href="<cfoutput>#AjaxLink('HelloWorld.cfm')#</cfoutput>">Click me</a>
</cfpod>
```
**AjaxOnLoad**

Description

Causes the specified JavaScript function to run when the page loads.

Returns

This function does not return a value.

Category

Display and formatting functions

Function syntax

```JavaScript
AjaxOnLoad(functionName)
```

See also

`cfdiv`, `cflayoutarea`, `cfpod`, `cfwindow`, Using Ajax User Interface Components and Features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>functionName</td>
<td>The name of the function to run when the page loads. The specified function does not take a parameter.</td>
</tr>
</tbody>
</table>

Usage

This function causes a JavaScript function to run when a page loads in the browser. The JavaScript function can perform any initialization actions that are required for the page to function properly. For example, a login window might open on a page if the user is not already logged in. You can use the `AjaxOnLoad` function to specify a JavaScript function that determines the login status and opens the window only if needed. You can use this function on top-level pages, or on pages that you dynamically include in your application by using the `source` attribute of the `cfpod` and `cfwindow` tags.

Example

The following example uses the `AjaxOnLoad` function to call an init function each time the page loads. The `init` function displays a login window.
<html>
<head>
<title>Enter Mail Login Details</title>
<script>
init = function() {
    ColdFusion.Window.show('loginwindow');
}
</script>
</head>
<body>
<cfwindow name="loginwindow" title="Enter Login Details"
          draggable="false" closable="false" resizable="false"
          width="450" height="200">
<cfoutput>
<form action="#cgi.script_name#" method="post" name="loginform">
<table width="400" class="loginTable" cellpadding="5">
<tr>
    <td style="text-align: right">mail server:</td>
    <td><input type="text" name="server"></td>
</tr>
<tr>
    <td style="text-align: right">username:</td>
    <td><input type="text" name="username"></td>
</tr>
<tr>
    <td style="text-align: right">password:</td>
    <td><input type="password" name="password"></td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td><input type="submit" name="login" value="Login"></td>
</tr>
</table>
</form>
</cfoutput>
<cfset AjaxOnLoad("init")>
</cfwindow>
</body>
</html>
ApplicationStop

Description

Stops or resets the current application. The application is restarted on the next request to the application.

Returns

void

Category

Other functions

Function Syntax

```
applicationstop()
```

History

Added in ColdFusion 9.

Example
<cftry>
...
</cftry>

<!--- stops the application to reset the artistid, if the artworks for the artist do not exist --->
<cfset applicationstop()/>
<cfset artist = EntityLoad( "artist", url.artistid, True )/>
<cfoutput>
<h2>#artist.getFullName()#</h2>
<table width="400">
<tr>
<th>Item</th>
<th>Price</th>
<th>Sold</th>
</tr>
<cfloop array="#artist.getArt()#" index="index">
<tr>
<td>#index.getArtName()#</td>
<td>#index.getPrice()#</td>
<td>#index.getIsSold()#</td>
</tr>
</cfloop>
</table>
<p><a href="index.cfm">View list</a></p>
</cfoutput>
<cfcatch type = "any">
<cfoutput>Artworks for the selected artists are not available</cfoutput>
</cfcatch>
</cftry>
ArrayAppend

Description

Appends an array element to an array.

Returns

True, on successful completion.

Category

Array functions

Function syntax

```
ArrayAppend(array, value [,merge])
```

See also

ArrayPrepend; Adding elements to an array in Basic array techniques in the Developing ColdFusion Applications

History

ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
<tr>
<td>arrayAppend</td>
<td>A boolean value that decides if array elements are added as one element at the end of the source array. By default, it would be false thereby retaining the old behavior.</td>
</tr>
<tr>
<td>value</td>
<td>Value to add at end of array</td>
</tr>
<tr>
<td>merge</td>
<td>If set to true appends array elements individually to the source array. If false (default) the complete array is added as one element at the end, in the source array.</td>
</tr>
</tbody>
</table>

Example
<h3>ArrayAppend Example</h3>
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
    SELECT FirstName, LastName FROM Employees
</cfquery>

<!--- Create an array --->
<cfset myArray = ArrayNew(1)>

<!--- Set element one to show where we are. --->
<cfset myArray[1] = "Test Value">

<!--- Loop through the query; append these names successively to the last element. --->
<cfloop query = "GetEmployeeNames">
    <cfoutput>
        Array was appended<br>
    </cfoutput>
</cfloop>

<!--- Show the resulting array as a list. --->
<cfset myList = ArrayToList(myArray, ",")>

<!--- Output the array as a list. --->
<cfoutput>
    The contents of the array are as follows:<p>
    #myList#</p>
</cfoutput>
ArrayAvg

Description

Calculates the average of the values in an array.

Returns

Number. If the array parameter value is an empty array, returns zero.

Category

Array functions, Mathematical functions

Function syntax

ArrayAvg(array)

See also

ArraySum; Basic array techniques in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
</tbody>
</table>

Usage

The following example uses the ColdFusion built-in variable Form.fieldNames, which is available on the action page of a form. It contains a comma-delimited list of the names of the fields on the form.

Example

```coldfusion
<!---- This example shows the use of ArrayAvg. --->
<!---- The following lines of code keep track of the form fields that can
be dynamically generated on the screen. It uses the Fieldnames variable
with the ListLen function to determine the number of fields on the form. --->
<cfset FormElem = 2>
<cfif Isdefined("Form.Submit")>
  <cfif Form.Submit is "More">
    <cfset FormElem = ListLen(Form.Fieldnames)>
  </cfif>
</cfif>

<html>
<head>
<title>ArrayAvg Example</title>
</head>
<body>
<h3>ArrayAvg Example</h3>
<p>This example uses ArrayAvg to find the average of the numbers that you enter
```
To enter more than two numbers click the <b>more</b> button.
</p>
<form action = "arrayavg.cfm">
<!--- The following code initially creates two fields. It adds fields if the user presses MORE. FormElem is initialized to two at the beginning of this code to show that the form has two fields. ----->
<input type = "submit" name = "submit" value = "more">
<table cellspacing = "2" cellpadding = "2" border = "0">
<cfloop index = "LoopItem" from = "1" to =="#FormElem#">
<tr>
<cfoutput>
<th align = "left">Number #LoopItem#</th>
<td><input type = "text" name = "number#LoopItem#"></td>
</cfoutput>
</tr>
</cfloop>
<input type = "submit" name = "submit" value = "get the average">
</table>
<!--- Create an array. --->
<cfif IsDefined("FORM.submit")>
<cfset myNumberArray = ArrayNew(1)>
<cfset Count = 1>
<cfloop index = "ListItem" list =="#Form.Fieldnames#">
<cfif Left(ListItem,3) is "Num">
<cfset myNumberArray[Count] = Val("number#Count#")>
<cfset count = IncrementValue(Count)>
</cfif>
</cfloop>
<cfif Form.Submit is "get the average">
<!--- use ArrayAvg to get the average of the two numbers --->
<p>The average of the numbers that you entered is</p>
<cfoutput>#ArrayAvg(myNumberArray)#.</cfoutput>
<cfelse>
<cfoutput>Try again. You must enter at least two numeric values.</cfoutput>
</cfoutput>
</cfif>
</cfif>
</body>
</html>
ArrayClear

Description

Deletes the data in an array.

Returns

True, on successful completion.

Category

Array functions

Function syntax

ArrayClear

See also

Array functions in Developing ColdFusion Applications

History

ColdFusion MX: Changed behavior: This function can be used on XML objects.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
</tbody>
</table>

Example

```c
foo = ["a", "b"];
arrayClear( foo );
// array foo is now empty
```
ArrayContains

Description

Description: Searches an array for the presence of a specified object. The function searches for simple objects such as strings and numbers or for complex objects such as structures. Simple object string searches are case-sensitive. This function does not support searches for COM and CORBA objects.

Returns

Yes, if the specified object exists in the array.

Category

Array functions

Function Syntax

ArrayContains(array, object)

See Also

ArrayFind, ArrayFindNoCase

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array to search in</td>
</tr>
<tr>
<td>object</td>
<td>Object to search for.</td>
</tr>
</tbody>
</table>

Example
<h3>ArrayContains Example</h3>

<!--Creating a 2-dimensional array-->
<cfset dayarray = ArrayNew(2)>
<cfset dayarray[1][1] = "Sunday">
<cfset dayarray[1][2] = "Monday">
<cfset dayarray[1][3] = "Tuesday">
<cfset dayarray[2][1] = "Wednesday">
<cfset dayarray[2][2] = "Thursday">
<cfset dayarray[2][3] = "Friday">

<cfoutput>
<p>Array contains</p>
#dayarray[1][1]#, #dayarray[1][2]#, #dayarray[1][3]#, #dayarray[2][1]#, #dayarray[2][2]#, #dayarray[2][3]#
<p>Checking value in the array</p>
#ArrayContains(dayarray[1], "Tuesday")#
</cfoutput>

<!--Creating a one-dimensional array-->
<h3>1-dimensional array example</h3>
<cfset montharray = ArrayNew(1)>
<cfset montharray[1] = "April">
<cfset montharray[2] = "July">
<cfset montharray[3] = "October">
<cfset montharray[4] = "December">

<p>Checking if value exists</p>
<cfoutput>
#ArrayContains(montharray, "December")#
</cfoutput>
ArrayDelete

Description

Searches an array for the first position of a specified object and deletes it. The function searches for simple objects such as strings and numbers or for complex objects such as structures. Simple object string searches are case-sensitive. This function does not support searches for COM and CORBA objects.

Returns

Yes, on successful deletion of the array element.

Category

Array functions

Function syntax

```
ArrayDelete(array, object)
```

See Also

ArrayDeleteAt, ArrayClear

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array to search in.</td>
</tr>
<tr>
<td>object</td>
<td>Object to search for.</td>
</tr>
</tbody>
</table>

Example

```
<cfscript>
    array = ["string","string"]; 
    writeOutput(arrayDelete(array, "string")); 
    writeDump(array);
</cfscript>
```
**ArrayDeleteAt**

**Description**

Deletes an element from a specified position in an array. When an element is deleted, ColdFusion recalculates index positions. For example, in an array that contains the months of the year, deleting the element at position 5 removes the entry for May. After this, to delete the entry for June, you would delete the element at position 5 (not 6).

**Returns**

True, on successful completion.

**Category**

Array functions

**Function syntax**

```
ArrayDeleteAt(array, position)
```

**See also**

*ArrayInsertAt*; Functions for XML object management in Modifying a ColdFusion XML object in the Developing ColdFusion Applications

**History**

ColdFusion MX:

- Changed behavior: This function can be used on XML objects.
- Changed thrown exceptions: This function can throw the InvalidArrayIndexException error.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
<tr>
<td>position</td>
<td>Array position</td>
</tr>
</tbody>
</table>

**Throws**

If this function attempts to delete an element at position 0, or specifies a value for `position` that is greater than the size of `array`, this function throws an InvalidArrayIndexException error.

**Example**


ArrayDeleteAt Example

<!---- Create an array. ---->
<cfset DaysArray = ArrayNew(2)>

<!---- Populate an element or two. ---->
<cfset DaysArray[1][1] = "Monday">
<cfset DaysArray[2][1] = "Tuesday">
<cfset DaysArray[3][1] = "Wednesday">
<cfset DaysArray[1][2] = "April 12">
<cfset DaysArray[2][2] = "April 13">
<cfset DaysArray[3][2] = "April 14">

This is what the array looks like before delete:<br>
<cfoutput>
#DaysArray[1][1]#&nbsp;&nbsp;#DaysArray[1][2]#<br>
#DaysArray[2][1]#&nbsp;&nbsp;#DaysArray[2][2]#<br>
#DaysArray[3][1]#&nbsp;&nbsp;#DaysArray[3][2]#<br>
</cfoutput>

We delete this element of the array:<br>
<ArrayDeleteAt(DaysArray,2)#<br>
</cfoutput>

<!--- The formerly third element, "Wednesday" is now the second element. --->
This is what the array looks like after delete:<br>
<cfoutput>
#DaysArray[1][1]#&nbsp;&nbsp;#DaysArray[1][2]#<br>
#DaysArray[2][1]#&nbsp;&nbsp;#DaysArray[2][2]#<br>
</cfoutput>
ArrayEach

Description

Used to loop over an array.

Returns

Nothing

Category

Closure functions

Syntax

arrayEach(array, function(any currentObj) {});

See also

Other closure functions.

History

ColdFusion 11: Passing array index to callback functions.

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of the array object.</td>
</tr>
<tr>
<td>function</td>
<td>Callback executed for each element in the array.</td>
</tr>
</tbody>
</table>

Passing array index to callback functions in ArrayEach

ColdFusion 10 supports passing objects in a callback function. From ColdFusion 11, you can pass the index of an array to the callback function as shown below:

<cfscript>

cityArray = ["San Jose","New york","Boston", "Las Vegas"];

function printArrayCity(city, index)
{
    writeOutput("<br>" & city & " is at index " & index);
}

ArrayEach(cityArray ,printArrayCity);

</cfscript>
Also, the original array can also be passed to the closure function. So the following code is also valid:

```cfcscript
// example 1 using an inline callback
numbers = [1,2,3,4,5];
doubledNumbers = [];
arrayEach(numbers, function(number){
    arrayAppend(doubledNumbers, number * 2);
});
writeOutput(arrayToList(doubledNumbers)); // outputs 2,4,6,8,10

// example 2, using a separately-defined function
numbers = [1,2,3,4,5];
doubledNumbers = [];
function double(number){
    arrayAppend(doubledNumbers, number * 2);
}
arrayEach(numbers, double); // here we use the NAME of an existing UDF
writeOutput(arrayToList(doubledNumbers)); // outputs 2,4,6,8,10
</cfcscript>
```
ArrayFilter

Description

Used to filter the elements of an array.

Returns

A new array

Category

Closure functions

Syntax

arrayFilter(array, function(arrayElement) {return true|false;});

Example

```cfscript
<cfscript>
names = ['Ray', 'Adam', 'Scott', 'Todd', 'Dave'];
shortNames = arrayFilter(names, function(n) {
  return len(n) < 4;
});
writeDump(shortNames);
</cfscript>
```

See also

Other closure functions.

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of the array object.</td>
</tr>
<tr>
<td>function</td>
<td>Inline function executed for each element in the array. Returns true if the array element has to be included in the resultant array.</td>
</tr>
<tr>
<td>arrayElement</td>
<td>Array element being accessed.</td>
</tr>
</tbody>
</table>
ArrayFind

Description

Searches an array for the first position of a specified object. The function searches for simple objects such as strings and numbers or for complex objects such as structures. When the second parameter is a simple object, string searches are case-sensitive. This function does not support searches for COM and CORBA objects.

Returns

Index in the array of the first match, or 0 if there is no match.

Category

Array functions

Function syntax

```
ArrayFind(array, object)
ArrayFind(array, function(arrayElement) {return true|false;})
```

See Also

ArrayFindNoCase

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array to search in.</td>
</tr>
<tr>
<td>function</td>
<td>Inline function executed for each element in the array. Returns true if the array element matches the search criterion.</td>
</tr>
<tr>
<td>object</td>
<td>Object to search for.</td>
</tr>
<tr>
<td>arrayElement</td>
<td>Array element being accessed.</td>
</tr>
</tbody>
</table>

Example

```
<cfscript>
    writeDump(ArrayFind(["STRING","string"], "string"));
    writeDump(ArrayFind(["STRING","string"], function(s) {if(compare(s, "string")==0) return true; return false;})�);
</cfscript>
```
ArrayFindAll

Description

Searches an array for all positions of a specified object. The function searches for simple objects such as strings and numbers or for complex objects such as structures. When the second parameter is a simple object, string searches are case-sensitive. This function does not support searches for COM and CORBA objects.

Returns

Array of indices in which the object was found.

Category

Closure functions

Syntax

ArrayFindAll(array, object)

ArrayFindAll(array, function(arrayElement) {return true|false;})

See also

Other closure functions.

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array to search in.</td>
</tr>
<tr>
<td>object</td>
<td>Object to search for.</td>
</tr>
<tr>
<td>function</td>
<td>Inline function executed for each element in the array. Returns true if the array elements match the search criterion.</td>
</tr>
<tr>
<td>arrayElement</td>
<td>Array element being accessed.</td>
</tr>
</tbody>
</table>

Example

<cfscript>
    writeDump(ArrayFindAll(["STRING","string"], "string"));
    writeDump(ArrayFindAll(["STRING","string"], function(s) {if(compare(s, "string")==0) return true; return false;}));
</cfscript>
ArrayFindAllNoCase

Description

Description: Searches an array for all positions of a specified object. The function searches for simple objects such as strings and numbers or for complex objects such as structures. When the second parameter is a simple object, string searches are not case-sensitive. This function does not support searches for COM and CORBA objects.

Note that ArrayFindAllNoCase() behaves identically to ArrayFindAll(), except in the special cases where the second parameter is a function or you are searching for a string in an array of strings. When the second parameter is a function, ArrayFindAll() executes the function for each element in the array and ArrayFindAllNoCase() searches the array for the function.

Returns

Array of indices in which the object was found.

Category

Closure functions

Syntax

```
ArrayFindAllNoCase(array, object)
```

See also

Other closure functions.

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array to search in.</td>
</tr>
<tr>
<td>object</td>
<td>Object to search for.</td>
</tr>
<tr>
<td>function</td>
<td>Inline function executed for each element in the array. Returns true if the array elements match the search criterion.</td>
</tr>
<tr>
<td>arrayElement</td>
<td>Array element being accessed.</td>
</tr>
</tbody>
</table>

Example
<cfscript>
    writeDump(ArrayFindAllNoCase(["STRING","string"], "string"));
</cfscript>
ArrayFindNoCase

Description

Searches an array for the first position of a specified object. The function searches for simple objects such as strings and numbers or for complex objects such as structures. When the second parameter is a simple object, string searches are not case-sensitive. This function does not support searches for COM and CORBA objects.

Note that ArrayFindNoCase() behaves identically to ArrayFind(), except in the special cases where the second parameter is a function or you are searching for a string in an array of strings. When the second parameter is a function, ArrayFind() executes the function for each element in the array and ArrayFindNoCase() searches the array for the function.

Returns

Index in the array of the first match, or 0 if there is no match.

Category

Array functions

History

ColdFusion 9: Added this function.

Function syntax

ArrayFindNoCase(array, object)

See Also

ArrayFind

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array to search in.</td>
</tr>
<tr>
<td>object</td>
<td>Object to search for</td>
</tr>
<tr>
<td>function</td>
<td>Inline function executed for each element in the array. Returns true if the</td>
</tr>
<tr>
<td>arrayElement</td>
<td>Array element being accessed.</td>
</tr>
</tbody>
</table>

Example
<cfscript>
    writeOutput(ArrayFindNoCase(["STRING","string"], "string");
</cfscript>
ArrayInsertAt

Description

Inserts a value into an array. Array elements whose indexes are equal to or greater than the new position are incremented by one. The array length increases by one.

Returns

True, on successful completion.

Category

Array functions

Function syntax

```
ArrayInsertAt(array, position, value)
```

See also

ArrayDeleteAt; Functions for XML object management in Modifying a ColdFusion XML object in the Developing ColdFusion Applications

History

ColdFusion MX:

- Changed behavior: This function can be used on XML objects.
- Changed thrown exceptions: This function can throw the InvalidArrayIndexException error.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
<tr>
<td>position</td>
<td>Index position at which to insert value</td>
</tr>
<tr>
<td>value</td>
<td>Value to insert</td>
</tr>
</tbody>
</table>

Usage

To apply the ArrayInsertAt() function to a multidimensional array, you must specify all but the last index in the array parameter. The following example inserts an element at myarray2:

```
<cfset ArrayInsertAt(myarray[2], 4, "test")>
```

Throws
If this function attempts to insert an element at position 0, or specifies a value for position that is greater than the size of array, this function throws an InvalidArrayIndexException error.

Example

```cfml
<cfset DaysArray = ArrayNew(1)>
<cfset DaysArray[1] = "Monday">
<cfset DaysArray[2] = "Tuesday">
<cfset DaysArray[3] = "Thursday">
<cfoutput>#ArrayInsertAt(DaysArray,3,"Wednesday")#</cfoutput>
<cfoutput>#ArrayToList(DaysArray)#</cfoutput>
```

Add an element before position 3:
```
<cfoutput>#ArrayInsertAt(DaysArray,3,"Wednesday")#</cfoutput>
```
Now output the array as a list:
```
<cfoutput>#ArrayToList(DaysArray)#</cfoutput>
```
```
ArrayIsDefined

Description

Determines whether an array element is defined.

Returns

True, if the array element is defined (exists); false, otherwise.

Category

Array functions

Function syntax

ArrayIsDefined(array, elementIndex)

See also

ArrayIsEmpty

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of a one-dimensional array, or the array name and indexes into higher-order dimensions of a multidimensional array.</td>
</tr>
<tr>
<td>elementIndex</td>
<td>Index of the element in a one-dimensional array, or the index of the element in the final dimension of a multidimensional array.</td>
</tr>
</tbody>
</table>

Usage

The index value of an element must be less than the length of the array. To test the existence of an element in a multidimensional array, specify all but the last dimension of the array in the first parameter. For example, the following line tests the existence of element MyArray[2][4][1]:

ArrayIsDefined(MyArray[2][4], 1)

Example
<h3>ArrayIsDefined Example</h3>

<!--- Create a sparse new array. --->
<cfset MyArray = ArrayNew(1)>

<!--- Populate an element or two. --->
<cfset MyArray[1] = "Test">
<cfset MyArray[3] = "Other Test">

<!--- Display the contents of the array. --->
<p>Your array contents are:
<cfdump var="#MyArray#"></p>

<!--- Check if an existing element is defined. --->
<p>Does element 3 exist?:
#ArrayIsDefined(MyArray, 3)#</p>

<!--- Check if a non-existent element is defined. --->
<p>Does element 2 exist?:
#ArrayIsDefined(MyArray, 2)#</p>
**ArrayIsEmpty**

**Description**

Determines whether an array is empty of data elements.

**Returns**

True, if the array is empty; False, otherwise.

**Category**

Array functions

**Function syntax**

```
ArrayIsEmpty(array)
```

**See also**

ArrayIsDefined, arrayLen, Functions for XML object management in Modifying a ColdFusion XML object in the Developing ColdFusion Applications

**History**

ColdFusion MX: Changed behavior: this function can be used on XML objects.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
</tbody>
</table>

**Usage**

You can test whether an element of a higher level dimension of a multidimensional array is empty by specifying the element in the ArrayIsEmpty function. To test whether the first row of the two-dimensional array MyArray is empty, use the following function:

```
ArrayIsEmpty(MyArray2[1])
```

**Example**
<h3>ArrayIsEmpty Example</h3>
<!--- Create a new array. --->
<cfset MyArray = ArrayNew(1)>
<!--- Populate an element or two. --->
<cfset MyArray[1] = "Test">
<cfset MyArray[2] = "Other Test">
<!--- Output the contents of the array. --->
<p>Your array contents are:
<cfoutput>#ArrayToList(MyArray)#</cfoutput>
<!--- Check to see if the array is empty. --->
<p>Is the array empty?:
<cfoutput>#ArrayIsEmpty(MyArray)#</cfoutput>
<!--- Now clear the array. --->
<cfset Temp = ArrayClear(MyArray)>
<!--- Check to see if the array is empty. --->
<p>Is the array empty?:
<cfoutput>#ArrayIsEmpty(MyArray)#</cfoutput>
arrayLen

Description

Determines the number of elements in an array.

Returns

The number of elements in an array.

Category

Array functions

Function syntax

arrayLen(array)

See also

ArrayIsEmpty; Functions for XML object management in Modifying a ColdFusion XML object in the Developing ColdFusion Applications

History

ColdFusion MX: Changed behavior: This function can be used on child XML objects.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
</tbody>
</table>

Example
<h3>arrayLen Example</h3>

```coldfusion
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
    SELECT FirstName, LastName FROM Employees
</cfquery>

<!--- Create an array. --->
<cfset myArray = arrayNew(1)>

<!--- Set element one to show where we are. --->
<cfset myArray[1] = "Test Value">

<!--- Loop through the query and append these names successively to the last element. --->
<cfloop query = "GetEmployeeNames">
    <cfset temp = ArrayAppend(myArray, "#FirstName# #LastName#")>
</cfloop>

<!--- Show the resulting array as a list. --->
<cfset myList = arrayToList(myArray, ",")>

<!--- Output the array as a list. --->
<cfoutput>
    <p>The contents of the array are as follows:</p>
    <p>#myList#</p>
    <p>This array has #arrayLen(MyArray)# elements.</p>
</cfoutput>
```
ArrayMap

Description

Iterates over every entry of the array and calls the closure function to work on the element of the array. The returned value will be set at the same index in a new array and the new array will be returned

Returns

Array

Syntax

```
ArrayMap(array, function(item [,index, array]))
```

History

ColdFusion 11: Added this function.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Required</td>
<td></td>
<td>The input array.</td>
</tr>
<tr>
<td>function</td>
<td>Required</td>
<td></td>
<td>Closure or a function reference that will be called for each of the iteration. The arguments passed to the callback are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• item: item in the array</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• index: index of the array</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• array: a reference of the original array</td>
</tr>
</tbody>
</table>
ArrayMax

Description

Array maximum function.

Returns

The largest numeric value in an array. If the array parameter value is an empty array, returns zero.

Category

Array functions

Function syntax

ArrayMax(array)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
</tbody>
</table>

Example

```cfcfml
t<!--- This example uses ArrayMax to find the largest number in an array. --->
<cfif IsDefined("FORM.submit")>
  <cfset myNumberArray = ArrayNew(1)>
  <cfset myNumberArray[1] = number1>
  <cfset myNumberArray[2] = number2>
  <cfif Form.Submit is "Maximum Value">
  <p>The largest number that you entered is</p>
  <cfoutput>#ArrayMax(myNumberArray)#</cfoutput>
  </cfif>
</cfif>
</cfcfml>
<cfif>
<!--- The following form provides two numeric fields that are compared when the form is submitted. --->
<form action = "arraymax.cfm">
  <input type = "hidden" name = "number1_Float">
  <input type = "hidden" name = "number2_Float">
  <input type = "text" name = "number1">
  <input type = "text" name = "number2">
  <input type = "submit" name = "submit" value = "Maximum Value">
</form>
```
ArrayMin

Description

Array minimum function.

Returns

The smallest numeric value in an array. If the array parameter value is an empty array, returns zero.

Category

Array functions

Function syntax

ArrayMin(array)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
</tbody>
</table>

Example

```<h3>ArrayMin Example</h3>
<p>This example uses ArrayMin to find the smallest number in an array.<br></p>
<!---- After checking whether the form has been submitted, this code creates an
array and assigns the form fields to the first two elements. ---->
<cfif IsDefined("FORM.submit")>
  <cfset myNumberArray = ArrayNew(1)>
  <cfset myNumberArray[1] = FORM.number1>
  <cfset myNumberArray[2] = FORM.number2>
  <cfif Form.Submit is "Minimum Value">
    <!--- Use ArrayMin to find the smallest number in the array. ---->
    <p>The smallest number that you entered is
    <cfoutput>#ArrayMin(myNumberArray)#.</cfoutput>
  </cfif>
</cfif>
<!---- The following form provides two numeric fields that are compared when the
form is submitted. ---->
<form action = "arraymin.cfm">
  <input type = "hidden" name = "number1_Float">
  <input type = "hidden" name = "number2_Float">
  <input type = "text" name = "number1"><br>
  <input type = "text" name = "number2"><br>
  <input type = "submit" name = "submit" value = "Minimum Value">
</form>```
**ArrayNew**

**Description**

Creates an array of 1-3 dimensions. Index array elements with square brackets: [ ]. ColdFusion arrays expand dynamically as data is added.

**Returns**

An array

**Category**

*Array functions*

**Function syntax**

```
ArrayNew(dimension)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimension</td>
<td>Number of dimensions in new array: 1, 2, or 3</td>
</tr>
</tbody>
</table>

**Example**

```
<h3>ArrayNew Example</h3>

<!-- Create an array. -->
<cfset MyNewArray = ArrayNew(1)>

<!-- ArrayToList does not function properly if the Array is not initialized with ArraySet -->
<cfset temp = ArraySet(MyNewArray, 1, 6, "")>

<!-- Set some elements. -->
<cfset MyNewArray[1] = "Sample Value">
<cfset MyNewArray[3] = "43">
<cfset MyNewArray[6] = "Another Value">

<!-- Is it an array? -->
<cfoutput>
  <p>Is this an array? #IsArray(MyNewArray)#</p>
  <p>It has #ArrayLen(MyNewArray)# elements.</p>
  <p>Contents: #ArrayToList(MyNewArray)#</p>

  <!-- The array has expanded dynamically to six elements with the use of ArraySet, even though we only set three values. -->
</cfoutput>
```
ArrayPrepend

Description
Inserts an array element at the beginning of an array.

Returns
True, on successful completion.

Category
Array functions

Function syntax

ArrayPrepend(array, value)

See also
ArrayAppend: Adding elements to an array in Basic array techniques in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
<tr>
<td>value</td>
<td>Value to insert at beginning of array</td>
</tr>
</tbody>
</table>

Example
<h3>ArrayPrepend Example</h3>
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
   SELECT FirstName, LastName FROM Employees
</cfquery>
<!--- Create an array. --->
<cfset myArray = ArrayNew(1)>
<!--- Set element one to show where we are. --->
<cfset myArray[1] = "Test Value">
<!--- Loop through query. Append names successively before last element. (The list reverses itself from the standard queried output, because it keeps prepending the array entry.) --->
<cfloop query = "GetEmployeeNames">
  <cfoutput>#ArrayPrepend(myArray, "#FirstName# #LastName#")#
</cfoutput>, Array was prepended<br>
</cfloop>
<!--- Show the resulting array as a list. --->
<cfset myList = ArrayToList(myArray, ",")>
<!--- Output the array as a list. --->
<cfoutput>
<p>The contents of the array are as follows:
<p>#myList#</cfoutput>
ArrayReduce

Description

Iterates over every entry of the array and calls the closure to work on the elements of the array. This function will reduce the array to a single value and will return the value.

Returns

any

Syntax

ArrayReduce(array, function(result, item, [,index, array])[, initialValue])

History

ColdFusion 11: Added this function.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Required</td>
<td></td>
<td>The input array.</td>
</tr>
<tr>
<td>function</td>
<td>Required</td>
<td></td>
<td>Closure or a function reference that will be called for each of the iteration. The arguments passed to the callback are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• result: result of the reduce operation after the previous iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• item: item in the array</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• index : current index for the iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• array : reference of the original array</td>
</tr>
<tr>
<td>initialValue</td>
<td>Optional</td>
<td></td>
<td>Initial value which will be used for the reduce operation. The type is any.</td>
</tr>
</tbody>
</table>

Example
<cfscript>
arr = [1,2,3,4,5];

function square(element, index)
{
    writeOutput("index is " & index);
    return element * element;
}

sq = arrayMap(arr, square);
writeDump(sq);

result = arrayReduce(sq, function(value, element)
{
    value = value?:0;
    value += element;
    return value;
});

writeDump(result);
</cfscript>
ArrayResize

Description

Resets an array to a specified minimum number of elements. Resetting can improve performance, if used to size an array to its expected maximum. For more than 500 elements, use `ArrayResize` immediately after using the `ArrayNew` tag. ColdFusion arrays expand dynamically as data is added.

Returns

True, on successful completion.

Category

Array functions

Function syntax

```
ArrayResize(array, minimum_size)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
<tr>
<td>minimum_size</td>
<td>Minimum array size</td>
</tr>
</tbody>
</table>

Example

```
<h3>ArrayResize Example</h3>
<cfquery name = "GetCourses" datasource = "cfdocexamples">
  SELECT * FROM Courses
</cfquery>
<cfset MyArray = ArrayNew(1)>
<cfset temp = ArrayResize(MyArray, GetCourses.RecordCount)>
<cfoutput>
  The array is now #ArrayLen(MyArray)# elements, to match
  the query of #GetCourses.RecordCount# records.
</cfoutput>
```
ArraySet

Description

In a one-dimensional array, sets the elements in a specified index range to a value. Useful for initializing an array after a call to ArrayNew.

Returns

True, on successful completion.

Category

Array functions

Function syntax

ArraySet(array, start_pos, end_pos, value)

See also

ArrayNew; Populating arrays with data in the Developing ColdFusion Applications

History

ColdFusion MX: Changed behavior: This function can be used on XML objects.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array.</td>
</tr>
<tr>
<td>start_pos</td>
<td>Starting index position of range to set.</td>
</tr>
<tr>
<td>end_pos</td>
<td>Ending index position of range to set. If this value is greater than array length, ColdFusion adds elements to array.</td>
</tr>
<tr>
<td>value</td>
<td>Value to which to set each element in the range.</td>
</tr>
</tbody>
</table>

Example
<h3>ArraySet Example</h3>

<!--- Create an array. --->
<cfset MyNewArray = ArrayNew(1)>
<!--- ArrayToList does not function properly if the Array has not been initialized with ArraySet. --->
<cfset temp = ArraySet(MyNewArray, 1,6, "Initial Value")>

<!--- Set some elements. --->
<cfset MyNewArray[1] = "Sample Value">
<cfset MyNewArray[3] = "43">
<cfset MyNewArray[6] = "Another Value">
...
ArraySlice

Description

Returns part of an array with only the elements you need.

Returns

Portion of an array based on the offset and length settings.

History

ColdFusion 10: Added this function.

Syntax

arraySlice(array,offset,length)

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of the array that you want to slice.</td>
</tr>
<tr>
<td>offset</td>
<td>Specifies the position from which to slice the array. Negative value indicates that the array is sliced, with sequence starting from array's end.</td>
</tr>
<tr>
<td>length</td>
<td>Element count from offset.</td>
</tr>
</tbody>
</table>

Example

```cfscript
<cfscript>
    array = [1, 2, 3, 4, 5, 6, 7, 8];
    newArray = arraySlice(array, 2, 3);//returns 2,3,4
    newArray = arraySlice(array, 4);//returns 4,5,6,7,8
    newArray = arraySlice(array, -5, 3);//returns 4,5,6
</cfscript>
```
ArraySort

Description
Sorts array elements numerically or alphanumerically.

Returns
True, if sort is successful; False, otherwise.

Category
Array functions, List functions

Function syntax

\[ \text{ArraySort(array, sort_type [, sort_order ])} \]

Or:

\[ \text{ArraySort(array, callback)} \]

History

ColdFusion 10:
- Added the localeSensitive attribute.
- Added alternative callback-based syntax

ColdFusion MX:
- Changed thrown exceptions: This function can throw the ArraySortSimpleValueException error and ValueNotNumeric error.
- Changed the order in which sorted elements are returned: In a textnocase, descending sort, this function might return elements in a different sort order than in earlier releases. If \( \text{sort_type} = "\text{textnocase}\) and \( \text{sort_order} = "\text{desc}\), ColdFusion processes elements that \textit{differ only in case}\) differently from earlier releases, as follows:
  - ColdFusion reverses the elements' original order.
  - Releases earlier than ColdFusion MX do not change the elements' original order.
  - For example, in a textnocase, desc sort of \( d, a, a, b, A\), the following occurs:
    - ColdFusion MX and later returns \( d, b, A, a, a\)
    - Releases earlier than ColdFusion MX return \( d, b, a, a, A\)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
</tbody>
</table>
localeSensitive | Specify if you wish to do a locale sensitive sorting. The default value is false.

sort_type | • numeric: sorts numbers
• text: sorts text alphabetically, taking case into account (also known as case sensitive). All letters of one case precede the first letter of the other case:
• aabzABZ, if sort_order = "asc" (ascending sort)- ZBAzbaa, if sort_order = "desc" (descending sort)
• textnocase: sorts text alphabetically, without regard to case (also known as case-insensitive). A letter in varying cases precedes the next letter:
• aAaBbBzzZ, in an ascending sort; preserves original intra-letter order - ZzzBbBaAa, in a descending sort; reverses original intra-letter order

sort_order | • asc - ascending sort order. Default.
• aabzABZ or aAaBbBzzZ, depending on value of sort_type, for letters- from smaller to larger, for numbers
• desc - descending sort order.
• ZBAzbaa or ZzzBbBaAa, depending on value of sort_type, for letters- from larger to smaller, for numbers

callback | A function which take two elements of the array, and returns whether the first is less than (-1), equal to (0) or greater than (1) the second one (similar to how compare() works for strings).

**Throws**

If an array element is something other than a simple element, this function throws an ArraySortSimpleValueException error. If sort_type is numeric and an array element is not numeric, this function throws a ValueNotNumeric error.

**Usage**

In ColdFusion 10, added support for all Java supported locale-specific characters (including support for umlaut characters). A flag for this support has been added for sorttype = "text" or sorttype = "textnocase".

**Example**
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
   SELECT FirstName, LastName FROM Employees
</cfquery>

<cfset myArray = ArrayNew(1)>

<cfloop query = "GetEmployeeNames">
<cfset temp = ArrayAppend(myArray, "#FirstName# #LastName#")>
</cfloop>

<cfset myList = ArrayToList(myArray, ",")>

<cfset isSuccessful = ArraySort(myArray, "textnocase", "desc")>

Example using a callback:

<cfscript>
authors = [
   {firstName="Witi", lastName="Ihimaera"},
   {firstName="Patricia", lastName="Grace"},
   {firstName="Alan", lastName="Duff"},
   {firstName="Lee", lastName="Tamahori"}, // OK: not an author
   {firstName="Keri", lastName="Hulme"} 
];
arraySort(
   authors,
   function (e1, e2){
      return compare(e1.lastName, e2.lastName);
   }
);
writeDump(authors);
</cfscript>

Note that the callback function does not need to be inline, as in the example; it can be any predefined user-defined function.
ArraySum

Description

Array sum function.

Returns

The sum of values in an array. If the array parameter value is an empty array, returns zero.

Category

Array functions, Mathematical functions

Function syntax

ArraySum(array)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
</tbody>
</table>

Example

```cfc
<h3>ArraySum Example</h3>
<p>This example uses ArraySum to add two numbers. </p>
<!--- After checking whether the form has been submitted, the code creates an array and assigns the form fields to the first two elements in the array. --->
<cfif IsDefined("FORM.submit")>
  <cfset myNumberArray = ArrayNew(1)>
  <cfset myNumberArray[1] = number1>
  <cfset myNumberArray[2] = number2>

  <cfif FORM.Submit is "Add">
    <!--- Use ArraySum to add the number in the array. --->
    <cfoutput>$ArraySum(myNumberArray)$</cfoutput>
  </cfif>
</cfif>

<!--- This form provides two numeric fields that are added when the form is submitted. --->
<form action = "arraysum.cfm" method="post">
  <input type = "hidden" name = "number1_Float">
  <input type = "hidden" name = "number2_Float">
  <input type = "text" name = "number1">
  <br>
  <input type = "text" name = "number2">
  <br>
  <input type = "submit" name = "submit" value = "Add">
</form>
```
ArraySwap

Description

Swaps array values of an array at specified positions. This function is more efficient than multiple `cfset` tags.

Returns

True, on successful completion.

Category

Array functions

Function syntax

```
ArraySwap(array, position1, position2)
```

See also

*Functions for XML object management* in *Modifying a ColdFusion XML object* in the *Developing ColdFusion Applications*

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of an array</td>
</tr>
<tr>
<td>position1</td>
<td>Position of first element to swap</td>
</tr>
<tr>
<td>position2</td>
<td>Position of second element to swap</td>
</tr>
</tbody>
</table>

Example

```
<h3>ArraySwap Example</h3>

<cfset month = ArrayNew(1)>
<cfset month[1] = "February">
<cfset month[2] = "January">
<cfset temp = ArraySwap(month, 1, 2)>
<cfset temp = ArrayToList(month)>

<p>Show the results: <cfoutput>#temp#</cfoutput>
```
ArrayToList

Description

Converts a one-dimensional array to a list.

Returns

Delimited list, as a string.

Category

Array functions, Conversion functions, List functions

Function syntax

ArrayToList(array [, delimiter ])

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Name of array</td>
</tr>
<tr>
<td>delimiter</td>
<td>Character or multicharacter string to separate list elements. The default value is comma.</td>
</tr>
</tbody>
</table>

Example
<h3>ArrayToList Example</h3>

<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
  SELECT FirstName, LastName FROM Employees
</cfquery>

<!--- Create an array. --->
<cfset myArray = ArrayNew(1)>

<!--- Loop through the query, append names successively to the last element. --->
<cfloop query = "GetEmployeeNames">
  <cfset temp = ArrayAppend(myArray, "#FirstName# #LastName#")>
</cfloop>

<!--- Show the resulting array as a list. --->
<cfset myList = ArrayToList(myArray, ",")>

<!--- Sort that array in descending order alphabetically. --->
<cfset myAlphaArray = ArraySort(myArray, "textnocase", "desc")>

<!--- Show the resulting alphabetized array as a list. --->
<cfset myAlphaList = ArrayToList(myArray, ",")>

<!--- Output the array as a list. --->
<cfoutput>
  <p>The contents of the array are as follows:  
  #myList#  
  This array, alphabetized by first name (descending):  
  #myAlphaList#  
  This array has #ArrayLen(MyArray)# elements.  
</cfoutput>
Asc

Description

Determines the value of a character.

Returns

The value of the first character of a string; if the string is empty, returns zero.

Category

String functions

Function syntax

```
Asc(string)
```

See also

Chr

History

ColdFusion MX: Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode characters, up to a value of 65536. (Earlier releases supported 1-255.)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string</td>
</tr>
</tbody>
</table>

Example
<h3>Asc Example</h3>

<!--- If the character string is not empty, output its ASCII value. --->
<cfif IsDefined("FORM.charVals")>

<!--- If FORM.charVals is not "" --->
<cfif FORM.charVals is not ">
  <cfoutput>
    #Left(FORM.charVals,1)# =
    #Asc(FORM.charVals)#
  </cfoutput>
</cfif>

<!--- If it is empty, output an error message. --->
<h4>Enter a character</h4>
</cfif>

<form action = "asc.cfm" method=post>
  Enter a character to see its ASCII value
  <br>
  <input type = "Text" name = "CharVals" size = "1" maxlength = "1">
  <input type = "Submit" name = "" > <input type = "RESET"/>
</form>
ASin

Description

Determines the arcsine of a number. The arcsine is the angle whose sine is \textit{number}.

Returns

The arcsine, in radians, of a number.

Category

Mathematical functions

Function syntax

\texttt{ASin(number)}

See also

Sin, Cos, ACos, Tan, Atn, Pi

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Sine of an angle. The value must be between -1 and 1, inclusive.</td>
</tr>
</tbody>
</table>

Usage

The range of the result is \(-\pi/2\) to \(\pi/2\) radians. To convert degrees to radians, multiply degrees by \(\pi/180\). To convert radians to degrees, multiply radians by \(180/\pi\).

Example
<h3>ASin Example</h3>

<!--- Output its arcsine value. --->
<cfif IsDefined("FORM.SinNum")>
  <cfif IsNumeric(FORM.SinNum)>
    <cfif FORM.SinNum LESS THAN OR EQUAL TO 1>
      <cfif FORM.SinNum GREATER THAN OR EQUAL TO -1>
        ASin(<cfoutput>#FORM.SinNum#/</cfoutput>) =
        <cfoutput>#ASin(FORM.SinNum)# Radians</cfoutput>
        <br> or <br>ASin(<cfoutput>#FORM.SinNum#/</cfoutput>) =
        <cfoutput>#ASin(FORM.SinNum) * 180/Pi()# Degrees</cfoutput>
      </cfif>
    </cfif>
  </cfif>
</cfif>

<!---- If it is less than negative one, output an error message. --->
<h4>Enter the sine of the angle to calculate, in degrees and radians. The value must be between 1 and -1, inclusive.</h4>
</cfif>

<!---- If it is greater than one, output an error message. --->
<h4>Enter the sine of the angle to calculate, in degrees and radians. The value must be between 1 and -1, inclusive.</h4>
</cfif>

<!---- If it is empty, output an error message. --->
<h4>Enter the sine of the angle to calculate, in degrees and radians. The value must be between 1 and -1, inclusive.</h4>
</cfif>
</cfif>

<form action = "./evaltest.cfm" method="post">
  Enter a number to get its arcsine in Radians and Degrees.
  <br><input type = "Text" name = "SinNum" size = "25">
  <input type = "Submit" name = ""> <input type = "RESET">
</form>
**Atn**

**Description**

Arctangent function. The arctangent is the angle whose tangent is *number*.

**Returns**

The arctangent, in radians, of a number.

**Category**

Mathematical functions

**Function syntax**

```
Atn(number)
```

**See also**

*Sin, ASin, Cos, ACos, Pi*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Tangent of an angle</td>
</tr>
</tbody>
</table>

**Usage**

The range of the result is -p/2 to p/2 radians. To convert degrees to radians, multiply degrees by p/180. To convert radians to degrees, multiply radians by 180/p.

**Example**

```
<h3>Atn Example</h3>
<!---- Output its Atn value. --->
<cffif IsDefined("FORM.AtnNum")>
  <cffif IsNumeric(FORM.AtnNum)>
    Atn(<cfoutput>#FORM.AtnNum#</cfoutput>) is
    <cfoutput>#Atn(FORM.AtnNum * 180/PI())# Degrees</cfoutput>
  <cfelse>
    <!---- If it is empty, output an error message. --->
    <h4>Enter a number</h4>
  </cffif>
</cffif>
</cffif>
<form action = "evaltest.cfm" method="post">
<p>Enter a number to get its arctangent in Radians and Degrees</p>
<br><input type = "Text" name = "atnNum" size = "25"></p>
<p><input type = "Submit" name = ""> <input type = "RESET"></p>
</form>
```
AuthenticatedContext

Description

This function is obsolete. Use the newer security tools; see Conversion functions and Securing Applications in the Developing ColdFusion Applications.

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later ColdFusion releases.
AuthenticatedUser

Description

This function is obsolete. Use the newer security tools; see Conversion functions and Securing Applications in the Developing ColdFusion Applications.

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later ColdFusion releases.
BinaryDecode

Description

Converts a string to a binary object. Used to convert binary data that has been encoded into string format back into binary data.

Returns

A binary object.

Category

Conversion functions, String functions

Function syntax

```
BinaryDecode(string, binaryencoding)
```

See also

BinaryEncode, CharsetEncode, CharsetDecode

History

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string containing encoded binary data.</td>
</tr>
<tr>
<td>binaryencoding</td>
<td>A string that specifies the algorithm used to encode the original binary data into a string; must be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• Hex: the characters 0-9 and A-F represent the hexadecimal value of each byte; for example, 3A.</td>
</tr>
<tr>
<td></td>
<td>• UU: data is encoded using the UNIX UUencode algorithm.</td>
</tr>
<tr>
<td></td>
<td>• Base64: data is encoded using the Base64 algorithm, as specified by IETF RFC 2045, at <a href="http://www.ietf.org/rfc/rfc2045.txt">www.ietf.org/rfc/rfc2045.txt</a>.</td>
</tr>
</tbody>
</table>

Usage

Use this function to convert a binary-encoded string representation of binary data back to a binary object for use in your application. Binary data is often encoded as a string for transmission over many Internet protocols, such as HTTP and SMTP, or for storage in a database. Adobe recommends that you use the BinaryDecode function, not the ToBinary(base64data) function, to convert Base64-encoded data to binary data in all new applications. See the following pages for additional information on handling binary data:
**Example**

The following example reads a GIF file as binary data, converts it to a binary-encoded string, converts the binary-encoded data back to binary data and writes the result to a file. It displays the encoded string and the image in the output file.

```cfml
<h3>Binary Encoding Conversion Example</h3>

<cfif IsDefined("Form.binEncoding")>
  <!--- Read in a binary data file. --->
  <cffile action="readbinary" file="C:\images\help.gif" variable="binimage">
  <!--- Convert the read data to binary encoding and back to binary data. --->
  <cfscript>
    binencode=BinaryEncode(binimage, Form.binEncoding);
    bindecode=BinaryDecode(binencode, Form.binEncoding);
  </cfscript>
  <!--- Write the converted results to a file. --->
  <cffile action="write" file="C:\temp\help.gif" output="#bindecode#" addnewline="No"/>
  <!--- Display the results. --->
  <cfoutput>
    <p>The binary encoding:</p>
    <h3>#Form.binEncoding#</h3>
    <p>The image converted into a binary-encoded string by BinaryEncode</p>
    <br>
    #binencode#
    <p>The image as written back to a file after converting back to binary using BinaryDecode</p>
    <br>
    <img src="C:\temp\help.gif"/>
  </cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
  Select binary encoding<br>
  <select size="1" name="binEncoding">
    <option selected>UU</option>
    <option>Base64</option>
    <option>Hex</option>
  </select><br>
  <input type = "Submit" value = "convert my data" />
</form>
```
BinaryEncode

Description

Converts binary data to a string.

Returns

An encoded string representing the binary data.

Category

Conversion functions, String functions

Function syntax

```
BinaryEncode(binarydata, encoding)
```

See also

BinaryDecode, CharsetEncode, CharsetDecode

History

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binarydata</td>
<td>A variable containing the binary data to encode.</td>
</tr>
<tr>
<td>encoding</td>
<td>A string that specifies the encoding method to use to represent the data; one of the following:</td>
</tr>
<tr>
<td></td>
<td>• Hex: use the characters 0-9 and A-F to represent the hexadecimal value of each byte; for example, 3A.</td>
</tr>
<tr>
<td></td>
<td>• UU: use the UNIX UUencode algorithm to convert the data.</td>
</tr>
<tr>
<td></td>
<td>• Base64: use the Base64 algorithm to convert the data, as specified by IETF RFC 2045, at <a href="http://www.ietf.org/rfc/rfc2045.txt">www.ietf.org/rfc/rfc2045.txt</a>.</td>
</tr>
</tbody>
</table>

Usage

Binary objects and, in some cases, 8-bit characters, cannot be transported over many Internet protocols, such as HTTP and SMTP, and might not be supported by some database systems. By Binary encoding the data, you convert the data into a format that you can transfer over any Internet protocol or store in a database as character data. To convert the data back to a binary format, use the BinaryDecode function. Adobe recommends that you use the BinaryEncode function, and not the ToBase64(binarydata) function, to convert binary data to Base64 data in all new applications. This function provides a superset of the functionality of the ToBase64(binarydata) function. See the following pages for additional information on handling binary data:
• **cffile** for loading and reading binary data
• **cfwddx** for serializing and deserializing binary data
• **IsBinary** for checking variables for binary format
• **Len** for determining the length of a binary object

Example

The following example reads a GIF file as binary data, converts it to a binary-encoded string, converts the binary-encoded data back to binary data, and writes the result to a file. It displays the encoded string and the image in the output file.

```html
<h3>Binary Encoding Conversion Example</h3>

<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.binEncoding")>

<!--- Read in a binary data file. --->
<cffile action="readbinary"
    file="C:\images\help.gif"
    variable="binimage">

<!--- Convert the read data to binary encoding and back to binary data. --->
<cfscript>
    binencode=BinaryEncode(binimage, Form.binEncoding);
    bindecode=BinaryDecode(binencode, Form.binEncoding);
</cfscript>

<!--- Write the converted results to a file. --->
<cffile action="write" file="C:\temp\help.gif" output="#bindecode#" addnewline="No">

<!--- Display the results. --->
<cfoutput>
    <p><b>The binary encoding:</b> #Form.binEncoding#</p>
    <p><b>The image converted into a binary-encoded string by BinaryEncode</b>
        #binencode#</p>
    <p><b>The image as written back to a file after converting back to binary</b>
        using BinaryDecode</p>
    <img src="C:\temp\help.gif">
</cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select binary encoding</b><br>
    <select size="1" name="binEncoding">
        <option selected>UU</option>
        <option>Base64</option>
        <option>Hex</option>
    </select><br>
    <input type = "Submit" value = "convert my data">
</form>
```
BitAnd

Description

Performs a bitwise logical AND operation.

Returns

The bitwise AND of two long integers.

Category

Mathematical functions

Function syntax

```
BitAnd(number1, number2)
```

See also

BitNot, BitOr, BitXor

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number1</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>number2</td>
<td>32-bit signed integer</td>
</tr>
</tbody>
</table>

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 - 2147483647.

Example

```
<h3>BitAnd Example</h3>

<p>Returns the bitwise AND of two long integers.</p>
<p>BitAnd(5,255): <cfoutput>#BitAnd(5,255)#</cfoutput></p>
<p>BitAnd(5,0): <cfoutput>#BitAnd(5,0)#</cfoutput></p>
<p>BitAnd(128,128): <cfoutput>#BitAnd(128,128)#</cfoutput></p>
```
BitMaskClear

Description

Performs a bitwise mask clear operation.

Returns

A number, bitwise cleared, with length bits beginning at start.

Category

Mathematical functions

Function syntax

BitMaskClear(number, start, length)

See also

BitMaskRead, BitMaskSet

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>start</td>
<td>Integer, in the range 0-31, inclusive; start bit for mask</td>
</tr>
<tr>
<td>length</td>
<td>Integer, in the range 0-31, inclusive; length of mask</td>
</tr>
</tbody>
</table>

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 - 2147483647.

Example

<h3>BitMaskClear Example</h3>

<p>Returns number bitwise cleared with length bits beginning from start.</p>

<p>BitMaskClear(255, 4, 4): <cfoutput>#BitMaskClear(255, 4, 4)#</cfoutput></p>
<p>BitMaskClear(255, 0, 4): <cfoutput>#BitMaskClear(255, 0, 4)#</cfoutput></p>
<p>BitMaskClear(128, 0, 7): <cfoutput>#BitMaskClear(128, 0, 7)#</cfoutput></p>
**BitMaskRead**

**Description**

Performs a bitwise mask read operation.

**Returns**

An integer, created from *length* bits of *number*, beginning at *start*.

**Category**

*Mathematical functions*

**Function syntax**

```
BitMaskRead(number, start, length)
```

**See also**

*BitMaskClear, BitMaskSet*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>32-bit signed integer to mask</td>
</tr>
<tr>
<td>start</td>
<td>Integer, in the range 0-31, inclusive; start bit for read</td>
</tr>
<tr>
<td>length</td>
<td>Integer, in the range 0-31, inclusive; length of mask</td>
</tr>
</tbody>
</table>

**Usage**

Bit functions operate on 32-bit signed integers, in the range -2147483648 - 2147483647.

**Example**

```
<h3>BitMaskRead Example</h3>
<p>Returns integer created from *length* bits of *number*, beginning with *start*.</p>

<p>BitMaskRead(255, 4, 4): <cfoutput>#BitMaskRead(255, 4, 4)#</cfoutput></p>
<p>BitMaskRead(255, 0, 4): <cfoutput>#BitMaskRead(255, 0, 4)#</cfoutput></p>
<p>BitMaskRead(128, 0, 7): <cfoutput>#BitMaskRead(128, 0, 7)#</cfoutput></p>
```
BitMaskSet

Description

Performs a bitwise mask set operation.

Returns

A number, bitwise masked with length bits of mask beginning at start.

Category

Mathematical functions

Function syntax

BitMaskSet(number, mask, start, length)

See also

BitMaskClear, BitMaskRead

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>mask</td>
<td>32-bit signed integer; mask</td>
</tr>
<tr>
<td>start</td>
<td>Integer, in the range 0-31, inclusive; start bit for mask</td>
</tr>
<tr>
<td>length</td>
<td>Integer, in the range 0-31, inclusive; length of mask</td>
</tr>
</tbody>
</table>

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 - 2147483647.

Example

```cfml
<h3>BitMaskSet Example</h3>
<p>Returns number bitwise masked with length bits of mask beginning at start.</p>

BitMaskSet(255, 255, 4, 4): <cfoutput>#BitMaskSet(255, 255, 4, 4)#</cfoutput></p>

BitMaskSet(255, 0, 4, 4): <cfoutput>#BitMaskSet(255, 0, 4, 4)#</cfoutput></p>

BitMaskSet(0, 15, 4, 4): <cfoutput>#BitMaskSet(0, 15, 4, 4)#</cfoutput></p>
```
**BitNot**

**Description**

Performs a bitwise logical NOT operation.

**Returns**

A number; the bitwise NOT of a long integer.

**Category**

Mathematical functions

**Function syntax**

```
BitNot(number)
```

**See also**

BitAnd, BitOr, BitXor

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>32-bit signed integer</td>
</tr>
</tbody>
</table>

**Usage**

Bit functions operate on 32-bit signed integers, in the range -2147483648 - 2147483647.

**Example**

```
<h3>BitNot Example</h3>
<p>Returns the bitwise NOT of a long integer.</p>
<p>BitNot(0): <cfoutput>#BitNot(0)#</cfoutput></p>
<p>BitNot(255): <cfoutput>#BitNot(255)#</cfoutput></p>
```
BitOr

Description

Performs a bitwise logical OR operation.

Returns

A number; the bitwise OR of two long integers.

Category

Mathematical functions

Function syntax

BitOr(number1, number2)

See also

BitAnd, BitNot, BitXor

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number1</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>number2</td>
<td>32-bit signed integer</td>
</tr>
</tbody>
</table>

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 - 2147483647.

Example

```html
<h3>BitOr Example</h3>
<p>Returns the bitwise OR of two long integers.</p>

<p>BitOr(5,255): <cfoutput>#BitOr(5,255)#</cfoutput></p>
<p>BitOr(5,0): <cfoutput>#BitOr(5,0)#</cfoutput></p>
<p>BitOr(7,8): <cfoutput>#BitOr(7,8)#</cfoutput></p>
```
BitSHLN

Description

Performs a bitwise shift-left, no-rotation operation.

Returns

A number, bitwise shifted without rotation to the left by count bits.

Category

Mathematical functions

Function syntax

BitSHLN(number, count)

See also

BitSHRN

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>count</td>
<td>Integer, in the range 0-31, inclusive; number of bits to shift the number</td>
</tr>
</tbody>
</table>

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 - 2147483647.

Example

```cfml
<h3>BitSHLN Example</h3>
<p>Returns the number, bitwise shifted, without rotation, to the left by count bits.</p>

<p>BitSHLN(1,1): <cfoutput>#BitSHLN(1,1)#</cfoutput></p>
<p>BitSHLN(1,30): <cfoutput>#BitSHLN(1,30)#</cfoutput></p>
<p>BitSHLN(1,31): <cfoutput>#BitSHLN(1,31)#</cfoutput></p>
<p>BitSHLN(2,31): <cfoutput>#BitSHLN(2,31)#</cfoutput></p>
```
**BitSHRN**

Description

Performs a bitwise shift-right, no-rotation operation.

Returns

A number, bitwise shifted, without rotation, to the right by *count* bits.

Category

**Mathematical functions**

Function syntax

```
BitSHRN(number, count)
```

See also

**BitSHLN**

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>count</td>
<td>Integer, in the range 0-31, inclusive. Number of bits to shift the number</td>
</tr>
</tbody>
</table>

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 - 2147483647.

Example

```
<h3>BitSHRN Example</h3>
<p>Returns a number, bitwise shifted, without rotation, to the right, by count bits.</p>
<p>BitSHRN(1,1): <cfoutput>#BitSHRN(1,1)#</cfoutput></p>
<p>BitSHRN(255,7): <cfoutput>#BitSHRN(255,7)#</cfoutput></p>
<p>BitSHRN(-2147483548,1): <cfoutput>#BitSHRN(-2147483548,1)#</cfoutput></p>
```
BitXor

Description

Performs a bitwise logical XOR operation.

Returns

Bitwise XOR of two long integers.

Category

Mathematical functions

Function syntax

BitXor(number1, number2)

See also

BitAnd, BitNot, BitOr

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number1</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>number2</td>
<td>32-bit signed integer</td>
</tr>
</tbody>
</table>

Usage

Bit functions operate on 32-bit signed integers, in the range -2147483648 - 2147483647.

Example

```html
<h3>BitXor Example</h3>
<p>Returns the bitwise XOR of two long integers.</p>
<p>BitXor(5,255): <cfoutput>#BitXor(5,255)#</cfoutput></p>
<p>BitXor(5,0): <cfoutput>#BitXor(5,0)#</cfoutput></p>
<p>BitXor(128,128): <cfoutput>#BitXor(128,128)#</cfoutput></p>
```

#back to top
ArrayDeleteNoCase

Description

Searches an array for the first position of a specified object and deletes it. The function searches for simple objects such as strings and numbers or for complex objects such as structures. Simple object string searches are case-insensitive. This function does not support searches for COM and CORBA objects.

Returns

Yes, on successful deletion of the array element.

Category

Array functions

Function syntax

```
ArrayDelete(array, object)
```

See Also

ArrayDeleteAt, ArrayClear

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Array to search in.</td>
</tr>
<tr>
<td>object</td>
<td>Object to search for.</td>
</tr>
</tbody>
</table>

Example

```
<cfscript>
    array = ["STRING","STRING"];  
    writeOutput(arrayDelete(array, "string"));
    writeDump(array);
</cfscript>
```
Functions c-d

- CacheGet
- CacheGetAllIds
- CacheGetMetadata
- CacheGetProperties
- cacheGetSession
- CacheIdExists
- CachePut
- CacheRegionExists
- CacheRegionNew
- CacheRegionRemove
- CacheRemove
- CacheRemoveAll
- CacheSetProperties
- CallStackDump
- CallStackGet
- CanDeSerialize
- Canonicalize
- CanSerialize
- Ceiling
- CharSetDecode
- CharSetEncode
- Chr
- CJustify
- Compare
- CompareNoCase
- Cos
- CreateDate
- CreateDateTime
- CreateObject
- CreateObject: .NET object
- CreateObject: COM object
- CreateObject: component object
- CreateObject: CORBA object
- CreateObject: Java or EJB object
- CreateObject: web service object
- CreateODBCDate
- CreateODBCDateTime
- CreateODBCTime
- CreateTime
- CreateTimeSpan
- CreateUUID
- CSRFGenerateToken
- CSRFVerifyToken
- DateAdd
- DateCompare
- DateConvert
- DateDiff
- DateFormat
- DatePart
- DateTimeFormat
- Day
- DayOfWeek
- DayOfWeekAsString
- DayOfYear
- DaysInMonth
- DaysInYear
- DE
- DecimalFormat
- DecodeForHTML
- DecodeFromURL
- DecrementValue
- Decrypt
- DecryptBinary
- DeleteClientVariable
- Deserialize
- DeserializeJSON
- DeserializeXML
- DirectoryCopy
- DirectoryCreate
- DirectoryDelete
- DirectoryExists
- DirectoryList
- DirectoryRename
- DollarFormat
- DotNetToCFType
- Duplicate
CacheIdExists

Description

Used to find if a cached object exists in the cache region. The region can be the default cache region (either at server or application level) or the custom region you specify.

Returns

True, if the cached object exists in the specified cache region.

History

ColdFusion 10: Added this function.

Syntax

cacheIdExists(id [, _ region___])

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>(Required) The ID of the cached object.</td>
</tr>
<tr>
<td>region</td>
<td>(Optional) The cache region where you check for the cached object.</td>
</tr>
</tbody>
</table>

Example: Checks if the cache object is present in the user-defined cache region

```cfml
<!--- Creating a new object ---
<cfset obj1 = structNew()>
<cfset obj1.name = "xyz">  

<!--- Defining the time to live and time to Idle parameters ---'
<cfset timeToLive=createtime(0,0,0,30)>  
<cfset timeToIdle=createtime(0,0,0,30)>  

<cfoutput>Starting to write to cache..</cfoutput>  
<cfset cachePut("obj1",obj1,timeToLive,timeToIdle,"customcache")>  
<br/>  
<cfoutput>Done!!</cfoutput>

<cfoutput>Trying to check if the cached item is present...</cfoutput>  
<cfoutput>#{cacheIdExists("obj1","customcache")}#</cfoutput>
```

Example: Checks if the cache object is present in the default cache region

```cfml
```
<cfset obj2 = structNew()>
  <cfset obj2.name = "xyz">
  <cfoutput>Starting to write to cache..</cfoutput>
  <cfset cachePut("obj2", obj2)>
  <br/>
  <cfoutput>Done!!</cfoutput>

  <cfoutput>Trying to fetch cached item...</cfoutput>
  <cfset obj = cacheGet("obj2")>
  <cfoutput>#cacheIdExists("obj2", "OBJECT")#</cfoutput>
CacheGet

Description

Gets an object that is stored in the cache.

Returns

The object stored in the cache.

Category

Cache functions

Function syntax

CacheGet(id, [region])

See also

cfcache, CachePut, CacheGetAllIds, CacheGetMetadata, CacheGetProperties, CacheRemove, CacheSetProperties

History

ColdFusion 10: Added the region parameter.ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID value assigned to the cache object when it was created.</td>
</tr>
<tr>
<td>region</td>
<td>(Optional) Specifies the cache region where you can place the cache object.</td>
</tr>
</tbody>
</table>
**CacheGetAllIds**

**Description**

Gets the IDs of all objects stored in the cache.

**Returns**

An array containing the IDs of all the objects that are stored in the cache.

**Category**

Cache functions

**Function syntax**

```coldfusion
CacheGetAllIds()
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>region</td>
<td>(Optional) The name of the cache region</td>
</tr>
<tr>
<td>accurate</td>
<td>(Optional) Boolean. When set to false, CacheGetAllIds will return the result faster. However, the result may not be accurate.</td>
</tr>
</tbody>
</table>

**See also**


**History**

ColdFusion 11: Added the attribute accurate

ColdFusion 10: Added the attribute region

ColdFusion 9: Added the function.
CacheGetMetadata

Description

Gets the metadata values for a cached object and template caches.

Returns

A structure containing the cached object metadata. The structure has the following fields:

<table>
<thead>
<tr>
<th>Structure element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache_hitcount</td>
<td>The number of cache entries have been used.</td>
</tr>
<tr>
<td>cache_misscount</td>
<td>Number of times the object was requested, but was not in the cache or was not up to date (was stale).</td>
</tr>
<tr>
<td>createdtime</td>
<td>The time when the element or object was cached.</td>
</tr>
<tr>
<td>hitcount</td>
<td>Number of times the cached object has been used.</td>
</tr>
<tr>
<td>idletime</td>
<td>The remaining time, in seconds, after which the cached object is purged if it is idle.</td>
</tr>
<tr>
<td>lasthit</td>
<td>The time the cached object was last used, as a date-time object, or the empty string.</td>
</tr>
<tr>
<td>lastupdated</td>
<td>The time the cached object was last modified, as a date-time object, or the empty string.</td>
</tr>
<tr>
<td>size</td>
<td>The number of bytes the object takes up when serialized.</td>
</tr>
<tr>
<td>timespan</td>
<td>The remaining time, in seconds, during which the cached object is valid.</td>
</tr>
</tbody>
</table>

Category

Cache functions

Function syntax

CacheGetMetadata(id, template)

See also

cfcache, CachePut, CacheGet, CacheGetAllIds, CacheGetProperties, CacheRemove, CacheSetProperties

History
ColdFusion 10: Added the attribute region
ColdFusion 9.0.1: Added the parameter `template`.
ColdFusion 9: Added the function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the cached object.</td>
</tr>
<tr>
<td>region</td>
<td>(Optional) The name of the cache region</td>
</tr>
<tr>
<td>template</td>
<td>Gets metadata for template caches.</td>
</tr>
</tbody>
</table>

**Example**

cacheTemp.cfm

```cfml
<cfcache action="cache" timespan="#CreateTimeSpan(0,0,10,0)#" useQueryString="true" metadata="cachemetadata">
<cfoutput>-- This date/time IS cached: #Now()#--<br /></cfoutput>
</cfcache>
```

getMetaDataTemplate.cfm

```cfml
<cfset baseurl = "http://#cgi.server_name#:#cgi.server_port##getDirectoryFromPath(cgi.script_name)#/cacheTemp.cfm">
<cfset pageid = hash(expandpath('cacheTemp.cfm'))>
<cfset lineno = 1>
<cfset templateCacheId = trim(ucase("#baseurl#_PAGEID:#pageid#_LINE:#lineno#"))>
<cfhttp url="#baseurl#" />
<cfset var="#getAllTemplateCacheIds()#"<br />
<cfset templateMetaData = cachegetmetadata(templatecacheid,'Template')>
<cfset var="#templateMetaData#">
```
cacheGetSession

Description

Lets you retrieve the underlying cache object to access additional cache functionality that is not implemented in the tag cfcache.

⚠️ Note

Caution Using the cacheGetSession function might pose security vulnerabilities. If you wish to disable the usage of this function, add it to Sandbox Security. For more information, see Using sandbox security.

Returns

The underlying cache object.

Syntax

cacheGetSession()

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectType</td>
<td>Any of the following values:</td>
</tr>
<tr>
<td></td>
<td>• object</td>
</tr>
<tr>
<td></td>
<td>• template</td>
</tr>
<tr>
<td></td>
<td>• name of the user-defined cache</td>
</tr>
<tr>
<td>isKey</td>
<td>Set to true if objectType is user-defined cache. The default value is false.</td>
</tr>
</tbody>
</table>

Category

Cache functions

See also

cfcache, CachePut, CacheGet, CacheGetAllIds, CacheGetMetadata, CacheRemove, CacheSetProperties

History

ColdFusion 9.0.1: Added this function

Example 1

The following example shows how to create a user-defined cache by adding an entry in ehCache.xml:
<cache
    name="customcache"
    maxElementsInMemory="1000"
    eternal="false"
    timeToIdleSeconds="720"
    timeToLiveSeconds="720"
    overflowToDisk="true"
    diskSpoolBufferSizeMB="10"
    maxElementsOnDisk="100000"
    diskPersistent="true"
    diskExpiryThreadIntervalSeconds="3600"
    memoryStoreEvictionPolicy="LRU"/>

Example 2

The following example shows how to use the function cacheGetSession to operate on default caches:
<!--- put an object into user-defined object cache --->
<cfset cachePut("cache1","hello",15,15)>

<!--- get underlying user-defined object cache --->
<cfset objectCache = cachegetsession("object",true)>

<!--- get/print user-defined object cache properties --->
<cfset config = objectCache.getCacheConfiguration()>
CachePut

Description

Stores an object in the cache.

Returns

Nothing

Category

Cache functions

Function syntax

CachePut(id, value, [timeSpan], [idleTime], [region], [throwOnError])

See also

cfcache, CacheGet, CacheGetAllIds, CacheGetProperties, CacheRemove, CacheSetProperties

History

ColdFusion 10: Added the region and throwOnError parameters. ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID for the cache object.</td>
</tr>
<tr>
<td>value</td>
<td>The value of the object. Can be any data type supported by ColdFusion</td>
</tr>
<tr>
<td>timeSpan</td>
<td>(Optional) The interval until the object is flushed from the cache, as a decimal number of days. One way to set the value is to use the return value from the CreateTimeSpan function. The default is to not time out the object.</td>
</tr>
<tr>
<td>idleTime</td>
<td>(Optional) A decimal number of days after which the object is flushed from the cache if it is not accessed during that time. One way to set the value is to use the return value from the CreateTimeSpan function.</td>
</tr>
<tr>
<td>region</td>
<td>Optional. Specifies the cache region where you can place the cache object.</td>
</tr>
<tr>
<td>throwOnError</td>
<td>Optional. If True and if regen does not exist, throws an error.</td>
</tr>
</tbody>
</table>
CacheRegionExists

Description

Checks if the cache region exists.

Returns

True if the cache region exists.

History

ColdFusion 10: Added this function.

Syntax

cacheRegionExists(region__)

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>region</td>
<td>Name of the cache region.</td>
</tr>
</tbody>
</table>

Example

```cfml
<!--- Checking if the region is present in the Cache --->
<cfif #cacheRegionExists("testregion")# EQ "YES">
   <cfset cacheRegionRemove("testregion")>
   <cfif #cacheRegionExists("testregion")# EQ "NO">
      <cfoutput>Region is deleted<br></cfoutput>
   </cfif>
</cfif>
<cfelse>
   <cfset cacheRegionNew("testregion")>
   <cfset cacheRegionRemove("testregion")>
   <cfif #cacheRegionExists("testregion")# EQ "NO">
      <cfoutput>Region is deleted<br></cfoutput>
   </cfif>
</cfif>
</cfif>
```
**CacheRegionNew**

**Description**

Creates a new custom cache region (if no cache region exists).

**Returns**

An error if `throwOnError` parameter is set to `true`, provided cache region already exists.

**History**

ColdFusion 10: Added this function.

**Syntax**

`cacheRegionNew(region,[properties],[throwOnError]___)`

**Properties**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>region</code></td>
<td>Name of the new cache region to be created.</td>
</tr>
<tr>
<td><code>properties</code></td>
<td>Optional. Struct that contains the cache region properties.</td>
</tr>
<tr>
<td><code>throwOnError</code></td>
<td>Optional. A Boolean value specifying if to throw an exception if the cache region name you specify already exists. The default value is <code>true</code>.</td>
</tr>
</tbody>
</table>

**Example**
<!--- Defining properties for the struct --->
<cfset defaultCacheProps = StructNew()>
<cfset defaultCacheProps.CLEARONFLUSH = "true">
<cfset defaultCacheProps.DISKEXPIRYTHREADINTERVALSECONDS = "3600">
<cfset defaultCacheProps.DISKPERSISTENT = "false">
<cfset defaultCacheProps.DISKSPOOLBUFFERSIZEMB = "30">
<cfset defaultCacheProps.ETERNAL = "false">
<cfset defaultCacheProps.MAXELEMENTSINMEMORY = "5">
<cfset defaultCacheProps.MAXELEMENTSONDISK = "10">
<cfset defaultCacheProps.MEMORYEVICTIONPOLICY = "LRU">
<cfset defaultCacheProps.OBJECTTYPE = "OBJECT">
<cfset defaultCacheProps.OVERFLOWTODISK = "true">
<cfset defaultCacheProps.TIMETOLIVESECONDS = "5">
<cfset defaultCacheProps.TIMETOIDLESECONDS = "30">
<cfset cacheRegionNew("testregion",#defaultCacheProps#,false)>

<cfset obj1 = structNew()>
<cfset obj1.name = "xyz">
<cfset timeToLive = CreateTimeSpan(0,0,5,0)>
<cfset timeToIdle = CreateTimeSpan(0,0,10,0)>

<cfoutput>Starting to write to cache..</cfoutput>
<cfset cachePut("obj1",obj1,timeToLive,timeToIdle,"testregion")>

<cfoutput>Trying to fetch cached item...</cfoutput>
<cfset obj = cacheGet("obj1","testregion")>
<br/>
<cfoutput>Done!!</cfoutput>
<cfoutput>#obj.name#</cfoutput>
CacheRegionRemove

Description

Removes a specified cache region.

Returns

Nothing

History

ColdFusion 10: Added this function.

Syntax

cacheRegionRemove(region__)

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>region</td>
<td>Name of the cache region that has to be removed.</td>
</tr>
</tbody>
</table>

Example

See the example for CacheRegionExists.
CacheRemove

Description

Removes an object from the cache.

Returns

Nothing.

Category

Cache functions

Function syntax

```
CacheRemove(ids, [throwOnError])
```

See also


History

ColdFusion 11: The ids parameter will also accept an array.

ColdFusion 10: Added the attributes region and {{exact}}.

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exact</td>
<td>(Optional) If true, the search narrows down to values that exactly match the IDs (for removal). The default value is true.</td>
</tr>
<tr>
<td>ids</td>
<td>A comma delimited string of IDs of the cached objects to remove. Starting from ColdFusion 11, you can also specify an array of IDs.</td>
</tr>
<tr>
<td>region</td>
<td>(Optional) Name of the cache region from which to remove the cached objects.</td>
</tr>
<tr>
<td>throwOnError</td>
<td>A Boolean value specifying whether to throw an exception if any ID does not specify a cached element. The default value is false.</td>
</tr>
</tbody>
</table>

Example with parameter exact set to true
```coldfusion
<!--- clear all object caches --->
<cfif ArrayLen(cacheGetAllIds()) gt 0>
    <cfset cacheRemove(ArrayToList(cacheGetAllIds()))>
</cfif>

<!--- create few caches --->
<cfloop from="1" to="10" index="i">
    <cfset id = "cache_#i#">
    <cfset timeToLive = CreateTimeSpan(0,0,30,0)>
    <cfset timeToIdle = CreateTimeSpan(0,0,30,0)>
    <cfset cachePut(id,createQryObj(i),timeToLive,timeToIdle)>
</cfloop>
<cfoutput>Before cacheRemove() :: Number of objects in the cache: #ArrayLen(cacheGetAllIds())#<br></cfoutput>

<!--- clear all objects from the cache --->
<cfset cacheRemove("cache_1",true,"object",true)>

<cfoutput>(List of cache Ids - #ListSort(ArrayToList(cacheGetAllIds()),"textnocase","ASC")#)</cfoutput><br>
<cfoutput>After cacheRemove() :: Number of objects in the cache: #ArrayLen(cacheGetAllIds())#<br><br></cfoutput>

Example with parameter exact set to false

<!--- create few caches --->
<cfloop from="1" to="10" index="i">
    <cfset id = "cache_#i#">
    <cfset timeToLive = CreateTimeSpan(0,0,30,0)>
    <cfset timeToIdle = CreateTimeSpan(0,0,30,0)>
    <cfset cachePut(id,createQryObj(i),timeToLive,timeToIdle)>
</cfloop>
<cfoutput>Before cacheRemove() :: Number of objects in the cache: #ArrayLen(cacheGetAllIds())#<br></cfoutput>

<!--- clear all objects from the cache --->
<cfset cacheRemove("cache",true,"object",false)>

<cfoutput>(List of cache Ids - #ListSort(ArrayToList(cacheGetAllIds()),"textnocase","ASC")#)</cfoutput><br>
<cfoutput>After cacheRemove() :: Number of objects in the cache: #ArrayLen(cacheGetAllIds())#<br><br></cfoutput>
```
CacheRemoveAll

Description

Removes all stored objects in a cache region. If no cache region is specified, objects in the default region are removed.

Returns

Nothing

History

ColdFusion 10: Added this function.

Syntax

`cacheRemoveAll(region__)`

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>region</td>
<td>(Optional) Indicates the cache region from which to remove the stored objects. If no value is specified, default cache region is considered by default.</td>
</tr>
</tbody>
</table>

Example

```cfml
<!--- clear all object caches --->
<cfif ArrayLen(cacheGetAllIds()) gt 0>
  <cfset cacheRemove(ArrayToList(cacheGetAllIds()))>
</cfif>

<!--- create few caches --->
<cfloop from="1" to="10" index="i">
  <cfset id = "cache_#i#">
  <cfset timeToLive = CreateTimeSpan(0,0,30,0)>
  <cfset timeToIdle = CreateTimeSpan(0,0,30,0)>
  <cfset cachePut(id,createQryObj(i),timeToLive,timeToIdle)>
</cfloop>

<!--- clear all objects from the cache --->
<cfset cacheRemoveAll()>

<!--- create few caches --->
<cfloop from="1" to="10" index="i">
  <cfset id = "cache_#i#">
  <cfset timeToLive = CreateTimeSpan(0,0,30,0)>
  <cfset timeToIdle = CreateTimeSpan(0,0,30,0)>
  <cfset cachePut(id,createQryObj(i),timeToLive,timeToIdle)>
</cfloop>
```

```
<!--- clear all object caches --->
<cfif ArrayLen(cacheGetAllIds()) gt 0>
  <cfset cacheRemove(ArrayToList(cacheGetAllIds()))>
</cfif>

<!--- create few caches --->
<cfloop from="1" to="10" index="i">
  <cfset id = "cache_#i#">
  <cfset timeToLive = CreateTimeSpan(0,0,30,0)>
  <cfset timeToIdle = CreateTimeSpan(0,0,30,0)>
  <cfset cachePut(id,createQryObj(i),timeToLive,timeToIdle)>
</cfloop>

<!--- clear all objects from the cache --->
<cfset cacheRemoveAll()>

<!--- create few caches --->
<cfloop from="1" to="10" index="i">
  <cfset id = "cache_#i#">
  <cfset timeToLive = CreateTimeSpan(0,0,30,0)>
  <cfset timeToIdle = CreateTimeSpan(0,0,30,0)>
  <cfset cachePut(id,createQryObj(i),timeToLive,timeToIdle)>
</cfloop>
```

```
CacheSetProperties

Description

Sets the cache properties for the object cache, the page cache, or both. If a cache does not yet exist, creates it. The cache and properties are application-specific.

Returns

Nothing

Category

Cache functions

Function syntax

```
CacheSetProperties(propStruct)
```

See also

cfcache, CacheGet, CachePut, CacheGetAllIds, CacheGetProperties, CacheGetMetadata, CacheRemove

History

ColdFusion 10: Added the attribute {{region}}ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propsStruct</td>
<td>A structure specifying the cache properties plus identifying information</td>
</tr>
<tr>
<td>region</td>
<td>(Optional) The name of the cache region</td>
</tr>
</tbody>
</table>

The `propsStruct` structure can have any or all of the following fields:

<table>
<thead>
<tr>
<th>Structure element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diskstore</td>
<td>The disk store.</td>
</tr>
<tr>
<td>diskpersistent</td>
<td>A Boolean value specifying whether to persist caches stored on disk through JVM restarts.</td>
</tr>
<tr>
<td>eternal</td>
<td>A Boolean value specifying whether no timeout or idletime applies. A true value indicates that the object or page is cached without any timespan being specified.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>maxelementsinmemory</td>
<td>The maximum number of objects that can be cached in memory. If the number is exceeded and <code>overflowtodisk</code> is false, the new objects entered replace old elements using algorithm specified in the <code>memoryevictionpolicy</code> entry.</td>
</tr>
<tr>
<td>maxelementsondisk</td>
<td>The maximum number of objects that can be stored on disk if <code>overflowtodisk</code> is true.</td>
</tr>
<tr>
<td>memoryevictionpolicy</td>
<td>The algorithm to used to evict old entries when maximum limit is reached, such as LRU (least recently used) or LFU (least frequently used).</td>
</tr>
<tr>
<td>objecttype</td>
<td>The cache type: <code>object</code> or <code>template</code>.</td>
</tr>
<tr>
<td>overflowtodisk</td>
<td>A Boolean value specifying whether when the maximum number of elements allowed in memory is reached, objects can be moved to disk, as determined by the <code>memoryevictionpolicy</code> value.</td>
</tr>
<tr>
<td>statistics</td>
<td><code>true</code> indicates that statistics collection for Ehcache is enabled.</td>
</tr>
<tr>
<td>timetoidoleconds</td>
<td>The idle time in seconds. Used if a <code>cfcache</code> tag does not specify an <code>idleTime</code> attribute.</td>
</tr>
<tr>
<td>timetolivesecond</td>
<td>The timeout time in seconds. Used if a <code>cfcache</code> tag does not specify a <code>timespan</code> attribute.</td>
</tr>
<tr>
<td>objecttype</td>
<td>The type of cache: one of the following:</td>
</tr>
<tr>
<td></td>
<td>• <code>template</code> - Set properties for the page cache, which contains cached pages and page segments.</td>
</tr>
<tr>
<td></td>
<td>• <code>object</code> - Set properties for the object cache.</td>
</tr>
<tr>
<td></td>
<td>• <code>all</code> - Set properties for both cache types.</td>
</tr>
</tbody>
</table>
CallStackGet

Description

Returns an array of structs. Each struct contains template name, line number, and if applicable the function name.

History

ColdFusion 10: Added this function.

Syntax

callStackGet(__)

Usage

Callstack is a snapshot of all function calls or invocations. For your ColdFusion application, callstack provides the template name, line number, and if applicable, the function name. The feature is helpful in scenarios where you want to track the recursive calls that you made. For example,

- a.cfm does an include of b.cfm.
- b.cfm invokes the function callStackGet at line 15.
  This results in a callstack that returns a struct that contains the template name, line number, and the function name.

Example

In this example, the factorial of a number is computed. Here,

1. The function callStackGet is called on line 9 in function factorial in the template fact.cfm
2. In turn, it is called from line 14 from the template fact.cfm
3. In turn, it is called from template callfact.cfm (line 2)
4. And so on, for other values of n

```cfml
<cftry>
  <cfinclude template="fact.cfm">
  <cfcatch type="any">
    <cfoutput>
      #cfcatch.message#
      <br>#cfcatch.detail#
    </cfoutput>
  </cfcatch>
</cftry>
```

fact.cfm
<cfscript>
numeric function factorial(n)
{
    if(n == 1)
    {
        writedump(callStackGet());
        writeoutput("<br>");
        return 1;
    }
    else
    {
        writedump(callStackGet());
        writeoutput("<br>");
        return n * factorial(n - 1);
    }
}
factorial(5);
</cfscript>
CallStackDump

Description

Similar to the function callStackGet except that it returns a string representation of the call stack.

History

ColdFusion 10: Added this function.

Syntax

callStackDump(destination__)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>Optional parameter that takes one of the following values:</td>
</tr>
<tr>
<td></td>
<td>• console</td>
</tr>
<tr>
<td></td>
<td>• browser: This is the default destination.</td>
</tr>
<tr>
<td></td>
<td>• file: If you do not provide the complete path to the file, the file is written to the temp directory as determined by the function getTempDirectory.</td>
</tr>
</tbody>
</table>

Usage

Callstack is a snapshot of all function calls or invocations. For your ColdFusion application, callstack provides the template name, line number, and if applicable, the function name. The feature is helpful in scenarios where you want to track the recursive calls that you made.

Example

In this example, the factorial of a number is computed. The example is similar to the example for CallStackGet except that the function used here is callStackDump.callfact.cfm

```cfc
cftry
cfinclude template="fact.cfm"
cfcatch type="any">
cfoutput
  #cfcatch.message#
<br>#cfcatch.detail#
<br>
</cfoutput>
cfcatch>
cftry>
```

fact.cfm
<cffunction name="factorial" hint="returns the factorial of a number" output="true">
    <cfargument name="n" required="yes" type="numeric" hint="The number for which the factorial is returned"/>
    <cfif n eq 1>
        <cfset callStackDump()>
        <cfreturn 1>
    <cfelse>
        <cfset callStackDump()>
        <cfreturn n * factorial(n - 1)>
    </cfif>
</cffunction>
<cfoutput>
    Factorial of 5 - #factorial(5)#
</cfoutput>
## CacheGetProperties

**Description**

Gets the cache properties for the object cache, the page cache, or both. The information is application-specific.

**Returns**

An Array of structures containing the cache properties. Each structure has the properties for the cache type: object or page. If you specify either type in the parameter, the array has a single structure entry. Each structure has the following fields:

<table>
<thead>
<tr>
<th>Structure element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diskpersistent</td>
<td>A Boolean value specifying whether to persist caches stored on disk through JVM restarts.</td>
</tr>
<tr>
<td>eternal</td>
<td>A Boolean value specifying whether no timeout or idletime applies. A true value indicates that the object or page is cached without any timespan being specified.</td>
</tr>
<tr>
<td>maxelementsinmemory</td>
<td>The maximum number of objects that can be cached in memory. If the number is exceeded and overflowtordisk is false, the new objects entered replace old elements using algorithm specified in the memoryevictionpolicy entry.</td>
</tr>
<tr>
<td>maxelementsondisk</td>
<td>The maximum number of objects that can be stored on disk if overflowtordisk is true.</td>
</tr>
<tr>
<td>memoryevictionpolicy</td>
<td>The algorithm to used to evict old entries when maximum limit is reached, such as LRU (least recently used) or LFU (least frequently used).</td>
</tr>
<tr>
<td>objecttype</td>
<td>The cache type: object or template.</td>
</tr>
<tr>
<td>overflowtordisk</td>
<td>A Boolean value specifying whether when the maximum number of elements allowed in memory is reached, objects can be moved to disk, as determined by the memoryevictionpolicy value.</td>
</tr>
<tr>
<td>statistics</td>
<td>True indicates that statistics collection for Ehcache is enabled.</td>
</tr>
<tr>
<td>timetoidolseconds</td>
<td>The idle time in seconds. Used if a cfcache tag does not specify an idletime attribute.</td>
</tr>
</tbody>
</table>
timetoliveseconds

The timeout time in seconds. Used if a cfcache tag does not specify a timespan attribute. Used if a cfquery tag does not specify a cachedwithin attribute.

The cacheGetProperties function also supports the following ehCache Structure Elements:

- cacheloadertimeoutmillis
- clearonflush
- copyonread
- copyonwrite
- diskaccessstripes
- diskexpirythreadintervalseconds
- diskpersistent
- diskspoolbuffersizemb
- eternal
- logging
- maxbyteslocaldisk
- maxbyteslocaldiskasstring
- maxbyteslocaldiskpercentageset
- maxbyteslocalheap
- maxbyteslocalheapasstring
- maxbyteslocalheappercentageset
- maxbyteslocaloffheap
- maxbyteslocaloffheapasstring
- maxbyteslocaloffheappercentageset
- maxelementsinmemory
- maxelementsondisk
- maxentrieslocaldisk
- maxentrieslocalheap
- maxmemoryoffheap
- maxmemoryoffheapinbytes
- memoryevictionpolicy
- name
- objecttype
- overflowtodisk
- overflowtooffheap
- overflowtooffheapset
- statistics
- timetoidleseconds
- timetoliveseconds

For more information on the above mentioned structure elements, refer to EhCache Documentation.

Category

Cache functions

Function syntax

```coldfusion
CacheGetProperties([type])
```

See also
cfcache, CachePut, CacheGet, CacheGetAllIds, CacheGetMetadata, CacheRemove, CacheSetProperties

History

ColdFusion 10: Added the attribute {{region}}ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| type      | (Optional) The cache type:  
|           | • template - Get properties for the page cache, which contains cached pages and page segments.  
|           | • object - Get properties for the object cache.  
|           | • no parameter - Get properties for both cache types.  
| region    | (Optional) The name of the cache region |
CanDeSerialize

Description

ColdFusion will invoke this function before invoking the Deserialize function.

Returns

Boolean. If true, the deserialize() function of the custom serializer will be called. If false, ColdFusion’s default deserializer will be used for serialization.

Function syntax

```cascading
cascading
    boolean function canDeSerialize(type)
cascading
```

History

ColdFusion 11: Added this function. For more information see REST Enhancements in ColdFusion 11.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Required. String. The type to which the data has to be deserialized.</td>
</tr>
</tbody>
</table>

Example

```xml
cfm
    <cffunction access="remote" name="canDeSerialize" returntype="boolean">
        <cfargument name="type" type="string"/>
        <!--- If you need to deserialize only XML using the custom serializer. --->
        <cfif #type# eq "XML">  
            <cfreturn true>
        <cfelse>
            <cfreturn false>
        </cfif>
    </cffunction>
```
Canonicalize

Description

Canonicalize or decode the input string.

Cross-site Scripting (XSS) is the most prevalent web application security flaw. This occurs when the user-supplied-data is sent to the browser without being properly validated. Canonicalization is the process of reducing a possibly encoded string down to its simplest form. Before validating any data, you must canonicalize the data. The canonicalize method can decode HTML entities, URL (Percent) encodings, and JavaScript encodings. In addition to simple decoding, canonicalize can also handle input, which is encoded using different techniques:

**Multiple Encoding**

By encoding the input multiple times or nesting the encoding using an encoding scheme.

Examples:

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; -&gt; &lt; -&gt; &amp;lt;&amp;#x3b</td>
<td>Encoded multiple times using the HTML Entity Encoding</td>
</tr>
<tr>
<td>&lt; -&gt; %3C -&gt; %253C -&gt; %25253C</td>
<td>Encoded multiple times using percent Encoding</td>
</tr>
<tr>
<td>\ -&gt; %5C -&gt; %33%63</td>
<td>Nested Encoding using URL (Percent) encoding multiple times</td>
</tr>
</tbody>
</table>

**Mixed Encoding**

When the input is encoded using different encoding schemes (For instance, encoded using both HTML and URL encoding).

Examples:

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; -&gt; &lt; -&gt; &amp;%6ct;</td>
<td>First encoded using HTML entity encoding and then encoded using the percent Encoding</td>
</tr>
<tr>
<td>&lt; -&gt; %3C -&gt; %&amp;x33;c</td>
<td>First encoded using URL (percent) encoding next nested encoded 3 using the HTML Entity encoding.</td>
</tr>
</tbody>
</table>

The data that is encoded more than once (nested or mixed) is something that normal users will not generate. Hence, having this kind of input data should be considered as malicious.

Returns

Decoded form of input string.

Category

Display and formatting functions

Syntax
canonicalize(inputString, restrictMultiple, restrictMixed [, throwOnError])

**History**

ColdFusion 11: Added the new attribute, throwOnError.

ColdFusion 10: Added this function.

**See also**

EncodeForHTML, EncodeForHTMLAttribute, EncodeForJavaScript, EncodeForCSS, EncodeForURL

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to be encode.</td>
</tr>
<tr>
<td>restrictMultiple</td>
<td>Required. If set to true, multiple encoding is restricted. This argument can be set to true to restrict the input if multiple or nested encoding is detected. If this argument is set to true, and the given input is multiple or nested encoded using one encoding scheme an error will be thrown.</td>
</tr>
<tr>
<td>restrictMixed</td>
<td>Required. If set to true, mixed encoding is restricted. This argument can be set to true to restrict the input if mixed encoding is detected. If this argument is set to true, and the given input is encoded using mixed encoding, an error will be thrown.</td>
</tr>
<tr>
<td>throwOnError</td>
<td>Optional. Default value is false. If restrictMultiple or restrictMixed is true and the given input contains mixed or multiple encoded strings, an exception will be thrown. If the value of this argument is true, instead of an exception, an empty string will be returned otherwise the exception will be thrown.</td>
</tr>
</tbody>
</table>

**Example**
<!---- canonicalize the simple html entity encoded string ---->
<cfoutput>#canonicalize("&lt;",false,false)#</cfoutput><br/>

<!---- enforce multiple and mixed encoding detection. Mixed encoding is detected as the data is encoded using URL and HTML entity encoding. Multiple Encoding is also detected ---->
<cftry>
<cfoutput>#canonicalize("%26lt; %26lt; %2526lt%253B %2526lt%253B %2526lt%253B",true,true, true)#</cfoutput><br/>
<cfcatch type="any" >
<!---- throws Error when throwOnError set to true when mixed or multiple encoding is detected. ---->
<cfdump var="#cfcatch#" >
</cfcatch>
</cftry>

<!---- enforce multiple and mixed encoding detection. Mixed encoding is detected as the data is encoded using URL and HTML entity encoding. Multiple Encoding is also detected ---->
<!---- an Empty string will be returned if the throwOnError is set to false and multiple or mixed encoding is found ---->
<cfoutput>#canonicalize("%26lt; %26lt; %2526lt%253B %2526lt%253B %2526lt%253B",false,true, false)#</cfoutput><br/>

<!---- enforce mixed but not multiple encoding detection returns an Empty String---->
<cfoutput>#canonicalize("%25 %2526 %26##X3c;script&##x3e; &##37;3Cscript%25252525253e",false,true)#</cfoutput><br/>

<cftry>
<cfoutput>#canonicalize("%26lt; %26lt; %2526lt%253B %2526lt%253B %2526lt%253B",false,true, true)#</cfoutput><br/>
<cfcatch type="any" >
<!---- throws Error when throwOnError set to true. ---->
<cfdump var="#cfcatch#" >
</cfcatch>
</cftry>

<!---- Mixed encoding is detected as the data is encoded using URL and HTML entity encoding. Multiple Encoding is also detected ---->
<!---- Decodes the string using both percent and HTML Entity encodings as the flags were set to false ---->
<cfoutput>#canonicalize("%26lt; %26lt; %2526lt%253B %2526lt%253B %2526lt%253B",false,false)#</cfoutput><br/>

<cfoutput>#canonicalize("%##X25;3c",false,false)#</cfoutput><br/>

<!---- Simple Javascript decoding ---->
<!---- http://www.planetpdf.com/codecuts/pdfs/tutorial/jsspec.pdf see section 2.7.5 for JS Encoding ---->
<cfoutput>#canonicalize("\U003C",false,false)#</cfoutput><br/>
<cfoutput>#canonicalize("\X3C",false,false)#</cfoutput><br/>
CanSerialize

Description

ColdFusion will invoke this function before invoking the serialize() function.

Returns

Boolean. If true, the serialize() function of the custom serializer will be called. If false, ColdFusion's default serializer will be used for serialization.

Function syntax

```java
boolean function canSerialize(type)
```

History

ColdFusion 11: Added this function. For more information see REST Enhancements in ColdFusion 11.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Required. String. The type to which the data has to be serialized.</td>
</tr>
</tbody>
</table>

Example

```xml
<cffunction access="remote" name="canSerialize" returntype="boolean">
    <cfargument name="type" type="string"/>
    <!--- If you need to serialize only XML using the custom serializer. --->
    <cfif #type# eq "XML">
        <cfreturn true>
    <cfelse>
        <cfreturn false>
    </cfif>
</cffunction>
```
Ceiling

Description

Determines the closest integer that is greater than a specified number.

Returns

The closest integer that is greater than a given number.

Category

Mathematical functions

Function syntax

Ceiling(number)

See also

Int, Fix, Round

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A real number</td>
</tr>
</tbody>
</table>

Example

```cftml
<h3>Ceiling Example</h3>
<cfoutput>
<p>The ceiling of 3.4 is #ceiling(3.4)#</p>
<p>The ceiling of 3 is #ceiling(3)#</p>
<p>The ceiling of 3.8 is #ceiling(3.8)#</p>
<p>The ceiling of -4.2 is #ceiling(-4.2)#</p>
</cfoutput>
```
**CharsetDecode**

**Description**

Converts a string to a binary representation.

**Returns**

A binary object that represents the string.

**Category**

Conversion functions, String functions

**Function syntax**

```
CharsetDecode(string, encoding)
```

**See also**

BinaryDecode, BinaryEncode, CharsetEncode; Determining the page encoding of server output in Processing a request in ColdFusion in the Developing ColdFusion Applications

**History**

ColdFusion MX 7: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string containing data to encode in binary format.</td>
</tr>
</tbody>
</table>
| encoding | A string that specifies encoding of the input data. Must be a character encoding name recognized by the Java runtime. The following list includes commonly used values:

- utf-8
- iso-8859-1
- windows-1252
- us-ascii
- shift_jis
- iso-2022- jp
- euc-jp
- euc-kr
- big5
- euc-cn
- utf-16

For a complete list of character encoding names supported by Sun Java runtimes, see [http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html](http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html) and [http://java.sun.com/j2se/1.4/docs/guide/intl/encoding.doc.html](http://java.sun.com/j2se/1.4/docs/guide/intl/encoding.doc.html). |

**Usage**

This function converts a string directly to a binary object. In releases of ColdFusion through ColdFusion MX 6.1, you had to use the `ToBase64` function to convert the string to Base64 and then use the `ToBinary` function to convert strings to binary data.

**Example**

The following example uses the `CharsetDecode` function to convert a string from a form to a binary object, and uses the `CharsetEncode` function to convert it back to the original value. You can change the character encoding that ColdFusion uses for the conversion. If you select the Asian language encodings, characters that are not in the specified character set are successfully converted.
<h3>Character Encoding Conversion Example</h3>
<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>

<!--- Do the conversions. --->
<cfscript>
    chardecode=CharsetDecode(Form.myString, Form.charEncoding);
    charencode=CharsetEncode(chardecode, Form.charEncoding);
</cfscript>

<!--- Display the input values and results. --->
<cfoutput>
    <!--- Parameter Settings --->
    <h3>Parameter Settings</h3>
    <p><b>The string:</b><br>
        #Form.myString#</p>
    <p><b>The character encoding:</b> #Form.charEncoding#</p>

    <!--- Results of the operations: --->
    <h3>Results of the operations:</h3>
    <p><b>Dump of the string converted to a binary object by CharsetDecode:</b><br>
        #chardecode#</p>
    <p><b>The binary object converted back to a string by CharsetEncode:<br>
        #charencode#</p>
</cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select the character encoding</b><br>
    <!--- This is a subset, additional encodings are available. --->
    <select size="1" name="charEncoding">
        <option selected>UTF-8</option>
        <option>ASCII</option>
        <option>ISO8859_1</option>
        <option>CP1252</option>
        <option>SJIS</option>
        <option>MS932</option>
        <option>EUC_CN</option>
        <option>Big5</option>
    </select><br>
    <br>
    <b>Enter a string</b><br>
    <textarea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">
    The following four characters are not in all character encodings:
    </textarea><br>
    <br>
    <input type = "Submit" value = "convert my data">
</form>
**CharsetEncode**

**Description**

Uses the specified encoding to convert binary data to a string.

**Returns**

A string representation of the binary object.

**Category**

Conversion functions, String functions

**Function syntax**

```plaintext
CharsetEncode(binaryobject, encoding)
```

**See also**

BinaryDecode, BinaryEncode, CharsetDecode; Determining the page encoding of server output in Processing a request in ColdFusion in the Developing ColdFusion Applications

**History**

ColdFusion MX 7: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binaryobject</td>
<td>A variable containing binary data to decode into text.</td>
</tr>
</tbody>
</table>
The character encoding that was used to encode the string into binary format. It must be a character encoding name recognized by the Java runtime. The following list includes commonly used values:

- utf-8
- iso-8859-1
- windows-1252
- us-ascii
- shift_jis
- iso-2022-jp
- euc-jp
- euc-kr
- big5
- euc-cn
- utf-16

For a complete list of character encoding names supported by Sun Java runtimes, see [http://java.sun.com/javase/6/docs/technotes/guides/intl/encoding.doc.html](http://java.sun.com/javase/6/docs/technotes/guides/intl/encoding.doc.html).

Usage

Adobe recommends that you use this function, and not the `ToString` function, to convert binary data to strings in all new applications.

Example

The following example uses the `CharsetDecode` function to convert a string from a form to a binary object, and uses the `CharsetEncode` function to convert it back to the original value. You can change the character encoding that ColdFusion uses for the conversion. If you select the Asian language encodings, characters that are not in the specified character set are successfully converted.
<h3>Character Encoding Conversion Example</h3>

<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>

<!--- Do the conversions. --->
<cfscript>
    chardecode=CharsetDecode(Form.myString, Form.charEncoding);
    charencode=CharsetEncode(chardecode, Form.charEncoding);
</cfscript>

<!--- Display the input values and results. --->
<cfoutput>
    <!--- Parameters --->
    <h3>Parameter Settings</h3>
    <p><b>The string:</b><br>
    #Form.myString#</p>
    <p><b>The character encoding:</b> #Form.charEncoding#</p>
    <h3>Results of the operations:</h3>
    <p><b>Dump of the string converted to a binary object by CharsetDecode:</b><br>
    </p>
    <cfdump var="#chardecode#"></p>
    <p><b>The binary object converted back to a string by CharsetEncode:</b><br>
    </p>
    #charencode#</p>
</cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select the character encoding</b><br>
    <!---- This is a subset, additional encodings are available. --->
    <select size="1" name="charEncoding" >
        <option selected>UTF-8</option>
        <option>ASCII</option>
        <option>ISO8859_1</option>
        <option>CP1252</option>
        <option>SJIS</option>
        <option>MS932</option>
        <option>EUC_CN</option>
        <option>Big5</option>
    </select><br>
    <br>
    <b>Enter a string</b><br>
    <textarea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">
The following four characters are not in all character encodings:
</textarea><br>
    <br>
    <input type = "Submit" value = "convert my data">
</form>
**Chr**

**Description**

Converts a numeric value to a UCS-2 character.

**Returns**

A character with the specified UCS-2 character code.

**Category**

String functions

**Function syntax**

```plaintext
Chr(number)
```

**See also**

Asc

**History**

ColdFusion MX: Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode characters, up to a value of 65535. (Earlier releases supported 1-255.)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A value (a number in the range 0 - 65535, inclusive)</td>
</tr>
</tbody>
</table>

**Usage**

The values 0 - 31 are standard, nonprintable codes. For example:

- `Chr(10)` returns a linefeed character
- `Chr(13)` returns a carriage return character
- The two-character string `Chr(13) & Chr(10)` returns a Windows newline

**Note**

For a complete list of the Unicode characters and their codes, see [www.unicode.org/charts/](http://www.unicode.org/charts/)

**Example**
<!--- If the character string is not empty, output its Chr value. --->
<cffif IsDefined("form.charVals")>
  <cfoutput>#form.charVals# = #Chr(form.charVals)#</cfoutput>
</cffif>

<cfform action="#CGI.script_name#" method="POST">
  <p>Type an integer character code from 1 to 65535<br>to see its corresponding character.<br>
  <cfinput type="Text"
    name="CharVals"
    range="1,65535"
    message="Enter an integer from 1 to 65535"
    validate="integer"
    required="Yes"
    size="5"
    maxlength="5"
  >
  </p>
  <input type="Submit" name="" > <input type="RESET">
</cfform>
CJustify

Description

Centers a string in a field length.

Returns

String, center-justified by adding spaces before or after the input parameter. If length is less than the length of the input parameter string, the string is returned unchanged.

Category

Display and formatting functions, String functions

Function syntax

Cjustify(string, length)

See also

LJustify, RJustify

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one. May be empty. If it is a variable that is defined as a number, the function processes it as a string.</td>
</tr>
</tbody>
</table>
| length    | A positive integer or a variable that contains one. Length of field. Can be coded as:  
  - A number; for example, 6  
  - A string representation of a number; for example, "6"  
    Any other value causes ColdFusion to throw an error. |

Example
<!-- This example shows how to use CJustify. -->
<CFPARAM name = "jstring" DEFAULT = "">

<cfif IsDefined("FORM.submit")>
<cfdump var="#Form#">
<cfset jstring = CJustify("#FORM.justifyString#", 35)>
</cfif>
<html>
<head>
<title>CJustify Example</title>
</head>
<body>
<h3>CJustify</h3>
<p>Enter a string; it will be center-justified within the sample field.</p>
<form action = "cjustify.cfm" method="post">
<p><input type = "Text" value = "<cfoutput>#jString#</cfoutput>
size = 35 name = "justifyString"> </p>
<p><input type = "Submit" name = "submit"> </p>
<input type = "RESET">
</form>
</body>
<html>
Compare

Description

Performs a case sensitive comparison of two strings.

Returns

- -1, if \textit{string1} is less than \textit{string2}
- 0, if \textit{string1} is equal to \textit{string2}
- 1, if \textit{string1} is greater than \textit{string2}

Category

\textit{String functions}

Function syntax

\texttt{Compare(string1, string2)}

See also

\texttt{CompareNoCase}, \texttt{Find}

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string1</td>
<td>A string or a variable that contains one</td>
</tr>
<tr>
<td>string2</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

Usage

Compares the values of corresponding characters in \textit{string1} and \textit{string2}.

Example
<h3>Compare Example</h3>
<p>The compare function performs a <i>case-sensitive</i> comparison of two strings.</p>

<cfif IsDefined("FORM.string1")>
<cfset comparison = Compare(FORM.string1, FORM.string2)>
<!--- Switch on the variable to give various responses. --->
<cfswitch expression = #comparison#>
<cfcase value = "-1">
<h3>String 1 is less than String 2</h3>
<i>The strings are not equal</i>
</cfcase>
<cfcase value = "0">
<h3>String 1 is equal to String 2</h3>
<i>The strings are equal!</i>
</cfcase>
<cfcase value = "1">
<h3>String 1 is greater than String 2</h3>
<i>The strings are not equal</i>
</cfcase>
<cfdefaultcase>
<h3>This is the default case</h3>
</cfdefaultcase>
<cfendswitch>
</cfif>
<form action = "compare.cfm" method="post">
<p>String 1</p>
<br><input type = "Text" name = "string1">
<p>String 2</p>
<br><input type = "Text" name = "string2">
<p><input type = "Submit" value = "Compare these Strings" name = ">
<input type = "RESET"></p>
</form>
## CompareNoCase

**Description**

Performs a case-insensitive comparison of two strings.

**Returns**

An indicator of the difference:

- A negative number, if `string1` is less than `string2`
- 0, if `string1` is equal to `string2`
- A positive number, if `string1` is greater than `string2`

**Category**

[String functions](#)

### Function syntax

```
CompareNoCase(string1, string2)
```

### See also

[Compare](#), FindNoCase; [Evaluation and type conversion issues in Data type conversion](#) in the Developing ColdFusion Applications

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string1</td>
<td>A string or a variable that contains one</td>
</tr>
<tr>
<td>string2</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

### Example
<H3>CompareNoCase Example</H3>

This function performs a <I>case-insensitive</I> comparison of two strings.

<cfif IsDefined("form.string1")>
<cfset comparison = CompareNoCase(form.string1, form.string2)>

<!--- switch on the variable to give various responses --->
<cfswitch expression=#comparison#>
  <cfcase value="-1">
    <H3>String 1 is less than String 2</H3>
    <I>The strings are not equal</I>
  </cfcase>
  <cfcase value="0">
    <H3>String 1 is equal to String 2</H3>
    <I>The strings are equal!</I>
  </cfcase>
  <cfcase value="1">
    <H3>String 1 is greater than String 2</H3>
    <I>The strings are not equal</I>
  </cfcase>
  <cfdefaultcase>
    <H3>This is the default case</H3>
  </cfdefaultcase>
</cfswitch>
</cfif>
<form action="comparenocase.cfm" method="POST">
  <P>String 1</P>
  <BR><input type="Text" name="string1">
  <P>String 2</P>
  <BR><input type="Text" name="string2">
  <P><input type="Submit" value="Compare these Strings" name="">
    <input type="RESET">
</form>
Cos

Description

Calculates the cosine of an angle that is entered in radians.

Returns

A number; the cosine of the angle.

Category

Mathematical functions

Function syntax

Cos(number)

See also

ACos, Sin, ASin, Tan, Atn, Pi

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Angle, in radians, for which to calculate the cosine</td>
</tr>
</tbody>
</table>

Usage

The range of the result is -1 to 1. To convert degrees to radians, multiply degrees by p/180. To convert radians to degrees, multiply radians by 180/p.

⚠️ Note

Because the function uses floating point arithmetic, it returns a small number (such as 6.1232399574E-017) for angles that must produce 0. To test for a 0 value, check whether the value is less than 0.0000000000001.

Example
<h3>Cos Example</h3>
<!--- Calculate cosine if form has been submitted. --->
<cfif IsDefined("FORM.cosNum")>
<!--- Make sure input is a number. --->
<cfif IsNumeric(#FORM.cosNum#)>
<!--- Convert degrees to radians, call the Cos function. --->
<cfset cosValue=#Cos((Form.cosNum * PI()) / 180)#>
<!--- 0.00000000000001 is the function's precision limit. If the absolute value
of the returned cosine value is less, set result to 0. --->
<cfif Abs(cosValue) LT 0.00000000000001>
<cfset cosValue=0>
</cfif>
<cfoutput>Cos(#FORM.cosNum#) = #cosValue#<br>
</cfoutput>
<!--- If input is not a number, show an error message. --->
</cfelse>
<h4>You must enter a numeric angle in degrees.</h4>
</cfif>
<form action="#CGI.script_name#" method="post">
Enter an angle in degrees to get its cosine:
<br><input type = "Text" name = "cosNum" size = "15">&nbsp;&nbsp;
<input type = "Submit" name = ">
<input type = "RESET" >
</form>
CreateDate

Description

Creates a date/time object.

Returns

A date/time value.

Category

Date and time functions

Function syntax

```
CreateDate(year, month, day)
```

See also

CreateDate, CreateODBCDateTime; Evaluation and type conversion issues in Data type conversion in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Integer in the range 0-9999. Integers in the range 0-29 are converted to 2000-2029. Integers in the range 30-99 are converted to 1930-1999. You cannot specify dates before AD 100.</td>
</tr>
<tr>
<td>month</td>
<td>Integer in the range 1 (January)-12 (December)</td>
</tr>
<tr>
<td>day</td>
<td>Integer in the range 1-31</td>
</tr>
</tbody>
</table>

Usage

CreateDate is a subset of CreateDateTime. The time in the returned object is set to 00:00:00.

Example
<h3>CreateDate Example</h3>
<cfif IsDefined("form.year")>
<p>Your date value, generated with CreateDate:</p>
<cfset yourDate = CreateDate(form.year, form.month, form.day)>
<cfoutput>
<ul>
<li>Formatted with CreateDate: #CreateDate(form.year, form.month, form.day)#</li>
<li>Formatted with CreateDateTime: #CreateDateTime(form.year, form.month, form.day, 12,13,0)#</li>
<li>Formatted with CreateODBCDate: #CreateODBCDate(yourDate)#</li>
<li>Formatted with CreateODBCDateTime: #CreateODBCDateTime(yourDate)#</li>
</ul>
<p>The same value can be formatted with DateFormat:</p>
<ul>
<li>Formatted with CreateDate and DateFormat: #DateFormat(CreateDate(form.year, form.month, form.day), "mmm-dd-yyyy")#</li>
<li>Formatted with CreateDateTime and DateFormat: #DateFormat(CreateDateTime(form.year, form.month, form.day, 12,13,0))#</li>
<li>Formatted with CreateODBCDate and DateFormat: #DateFormat(CreateODBCDate(yourDate), "mmm d, yyyy")#</li>
<li>Formatted with CreateODBCDateTime and DateFormat: #DateFormat(CreateODBCDateTime(yourDate), "d/m/yy")#</li>
</ul>
</cfoutput>
</cfif>
</cfif>
</cfform>
CreateDateTime

Description

Creates a date-time object.

Returns

A date/time value.

Category

Date and time functions

Function syntax

CreateDateTime(year, month, day, hour, minute, second)

See also

CreateDate, CreateTime, CreateODBCDateTime, Now; Evaluation and type conversion issues in Data type conversion in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Integer in the range 0-9999. Integers in the range 0-29 are converted to 2000-2029. Integers in the range 30-99 are converted to 1930-1999. You cannot specify dates before AD 100.</td>
</tr>
<tr>
<td>month</td>
<td>Integer in the range 1 (January) - 12 (December)</td>
</tr>
<tr>
<td>day</td>
<td>Integer in the range 1 - 31</td>
</tr>
<tr>
<td>hour</td>
<td>Integer in the range 0 - 23</td>
</tr>
<tr>
<td>minute</td>
<td>Integer in the range 0 - 59</td>
</tr>
<tr>
<td>second</td>
<td>Integer in the range 0 - 59</td>
</tr>
</tbody>
</table>

Example
<h3>CreateDateTime Example</h3>

<cfif IsDefined("form.year")>
Your date value, generated with CreateDateTime:
<cfset yourDate = CreateDateTime(form.year, form.month, form.day,
form.hour, form.minute, form.second)>

<cfoutput>
<ul>
<li>Formatted with CreateDate: #CreateDate(form.year, form.month, form.day)#</li>
<li>Formatted with CreateDateTime: #CreateDateTime(form.year, form.month, form.day, form.hour, form.minute, form.second)#</li>
<li>Formatted with CreateODBCDate: #CreateODBCDate(yourDate)#</li>
<li>Formatted with CreateODBCDateTime: #CreateODBCDateTime(yourDate)#</li>
</ul>
<p>The same value can be formatted with DateFormat:
<ul>
<li>Formatted with CreateDate and DateFormat:
 #DateFormat(CreateDate(form.year, form.month, form.day), "mmm-dd-yyyy")#</li>
<li>Formatted with CreateDateTime and DateFormat:
 #DateFormat(CreateDateTime(form.year, form.month, form.day, form.hour, form.minute, form.second))#</li>
<li>Formatted with CreateODBCDate and DateFormat:
 #DateFormat(CreateODBCDate(yourDate), "mmm d, yyyy")#</li>
<li>Formatted with CreateODBCDateTime and DateFormat:
 #DateFormat(CreateODBCDateTime(yourDate), "d/m/yy")#</li>
</ul>
</cfoutput>
</cfif>

<cfform action="createdatetime.cfm" method="post">
<p>Please enter the year, month, and day, in integer format, for a date to view:
<pre>
Year <cfinput type="text" name="year" value="1998" validate="integer"
required="yes" />
Month <cfinput type="text" name="month" value="6" range="1,12"
message="Please enter a month (1-12)" validate="integer"
required="yes" />
Day <cfinput type="text" name="day" value="8" range="1,31"
message="Please enter a day of the month (1-31)" validate="integer"
required="yes" />
Hour <cfinput type="text" name="hour" value="16" range="0,23"
message="You must enter an hour (0-23)" validate="integer"
required="yes" />
Minute <cfinput type="text" name="minute" value="12" range="0,59"
message="You must enter a minute value (0-59)" validate="integer"
required="yes" />
Second <cfinput type="text" name="second" value="0" range="0,59"
message="You must enter a value for seconds (0-59)" validate="integer"
required="yes" />
</pre>
<p><input type="submit" name="" />
<input type="reset">
</cfform>
CreateObject

Description

Creates a ColdFusion object, of a specified type.

Returns

An object, of the specified type.

Category

Extensibility functions

History

ColdFusion 10: Added the parameter wsVersion ColdFusion 9:

- ColdFusion 9 does not require the type argument.

ColdFusion 8:

- Added the .NET/dotnet type
- For web service object, added the WSDL2Java and argstruct parameters

ColdFusion MX 7: For web service object: added the portName parameter, which specifies a port named in the service element of the WSDL.

ColdFusion MX:

1. Changed instantiation behavior: this function, and the cfobject tag, can instantiate ColdFusion components and web services. Executing operations on a CFC object executes CFML code that implements the CFC's method in the CFC file. For more information, see the Developing ColdFusion Applications.
2. For CORBA object: changed the Naming Service separator format for addresses from a dot to a forward slash. For example, if "context=NameService", for a class, use either of the following formats for the class parameter:
   - "/Eng/CF"
   - ".current/Eng.current/CF"
   (In earlier releases, the format was ".Eng.CF").
1. For CORBA object: changed the locale parameter; it specifies the Java config that contains the properties file.

CreateObject object types

For information about using this function, see these sections:

- CreateObject: .NET object
- CreateObject: COM object
- CreateObject: component object
- CreateObject: CORBA object
- CreateObject: Java or EJB object
- CreateObject: web service object

⚠️ Note

On UNIX, this function does not support COM objects.
CreateObject: .NET object

Description

Creates a .NET object, that is, a ColdFusion proxy for accessing a class in a local or remote .NET assembly.

Returns

A .NET object, that is, a ColdFusion reference to a local or remote .NET assembly class.

Function syntax

```
CreateObject(type, class, assembly[, server, port, protocol, secure])
```

See also

DotNetToCFType, Using Microsoft .NET Assemblies in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>component</td>
<td>Object type. Must be .NET or dotnet for .NET objects.</td>
</tr>
<tr>
<td>class</td>
<td></td>
<td>Name of the .NET class to represent as an object.</td>
</tr>
</tbody>
</table>
### assembly

<table>
<thead>
<tr>
<th>assembly</th>
<th>mscorlib.dll which contains the .NET core classes</th>
</tr>
</thead>
</table>

For local .NET assemblies, the absolute path or paths to the assembly or assemblies (.exe or .dll files) from which to access the .NET class and its supporting classes. If a class in an assembly requires supporting classes that are in other assemblies, specify those assemblies also. You can, however, omit the supporting assemblies for the following types of supporting classes:

- .NET core classes (classes in mscorlib.dll)
- classes in assemblies that are in the global assembly cache (GAC)

To specify multiple assemblies, use a comma-delimited list.

For remote .NET assemblies, specify the absolute path or paths of the local proxy JAR file or files that represent the assemblies. If you omit this parameter, and there is no local .NET installation, the function fails without generating an error. If you omit this parameter, there is a local .NET installation, and the specified class is not in the .NET core classes, ColdFusion generates an error.

### server

<table>
<thead>
<tr>
<th>server</th>
<th>localhost</th>
</tr>
</thead>
</table>

Host name or IP address of the server where the .NET-side agent is running. Can be in any of these forms:

- server name (for example, myserver)
- IP address (for example, 127.0.0.1)

Specify this attribute to access .NET components on a remote server.

### port

<table>
<thead>
<tr>
<th>port</th>
<th>6086</th>
</tr>
</thead>
</table>

Port number at which the .NET-side agent is listening.
| **protocol** | tcp | Protocol to use for communication between ColdFusion and .NET. Must be one of the following values:

- **http**: Use HTTP/SOAP communication protocol. This option is slower than tcp, but might be required for access through a firewall.
- **tcp**: Use binary TCP/IP protocol. This method is more efficient than HTTP.

| **secure** | false | Whether to secure communications with the .NET-side agent. If true, ColdFusion uses SSL to communicate with .NET.

**Usage**

The `CreateObject` function and `cfobject` tag differ only in syntax. For more information on creating ColdFusion .NET objects. For detailed information on using the .NET assemblies in ColdFusion, see [Using Microsoft .NET Assemblies](#) in the *Developing ColdFusion Applications*.
CreateObject: COM object

Description

The CreateObject function can create a Component Object Model (COM) object. To create a COM object, provide the following information:

- The object's program ID or filename
- The methods and properties available to the object through the IDispatch interface
- The arguments and return types of the object's methods

For most objects, you can get this information from the OLEView utility.

⚠️ Note

On UNIX, this function does not support COM objects.

Returns

A COM object.

Function syntax

CreateObject(type, class, context, serverName)

See also

ReleaseComObject, cfobject; Integrating COM and CORBA Objects in CFML Applications in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| type      | Type of object to create.  
- com  
- corba  
- java  
- component  
- webservice  
The default value of type is component. |
| class     | Component ProgID for the object to invoke. |
| context   |  
- InProc  
- Local  
- Remote |
serverName

Server name, using UNC or DNS convention, in one of these forms:

- lanserver
- lanserver
- http://www.servername.com
- www.servername.com
- 127.0.0.1

If context = "remote", this parameter is required.

Usage

The following example creates the Windows Collaborative Data Objects (CDO) for NTS NewMail object to send mail. You use this code in a cfscript tag.

```
Mailer = CreateObject("COM", "CDONTS.NewMail");
```
**CreateObject: component object**

**Description**

The **CreateObject** function can create an instance of a ColdFusion component (CFC) object.

**Returns**

A component object.

**Function syntax**

```
CreateObject(type, component-name)
```

**See also**

[Building and Using ColdFusion Components](Developing%20ColdFusion%20Applications) in the *Developing ColdFusion Applications*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>type</strong></td>
<td>Type of object to create.</td>
</tr>
<tr>
<td></td>
<td>• com</td>
</tr>
<tr>
<td></td>
<td>• corba</td>
</tr>
<tr>
<td></td>
<td>• java</td>
</tr>
<tr>
<td></td>
<td>• component</td>
</tr>
<tr>
<td></td>
<td>• webservice</td>
</tr>
<tr>
<td></td>
<td>The default value of type is component.</td>
</tr>
<tr>
<td><strong>component-name</strong></td>
<td>The CFC name; corresponds to the name of the file that defines the component; for example, use engineComp to specify the component defined in the engineComp.cfc file</td>
</tr>
</tbody>
</table>

**Usage**

On UNIX systems, ColdFusion searches first for a file with a name that matches the specified component name, but is all lowercase. If it does not find the file, it looks for a filename that matches the component name exactly, with the identical character casing. In the following example, the CFScript statements assign the `tellTimeCFC` variable to the `tellTime` component using the **CreateObject** function. The **CreateObject** function references the component in another directory. To invoke component methods, you use function syntax.
<b>Server's Local Time:</b>
<cfscript>
tellTimeCFC=CreateObject("component","appResources.components.tellTime");
tellTimeCFC.getLocalTime();
</cfscript>
<br>
<b>Calculated UTC Time:</b>
<cfscript>
tellTimeCFC.getUTCTime();
</cfscript>
CreateObject: CORBA object

Description

The CreateObject function can call a method on a CORBA object. The object must be defined and registered for use.

Returns

A handle to a CORBA interface.

Function syntax

```plaintext
CreateObject(type, context, class, locale)
```

See also

Integrating COM and CORBA Objects in CFML Applications in the Developing ColdFusion Applications

History

See the History section of the main CreateObject function page.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Type of object to create.</td>
</tr>
</tbody>
</table>
| context   | • IOR: ColdFusion uses IOR to access CORBA server  
|           | • NameService: ColdFusion uses naming service to access server. Valid only with the InitialContext of a VisiBroker ORB. |
| class     | • If context = "ior": absolute path of file that contains string version of the Interoperable Object Reference (IOR). ColdFusion must be able to read file; it must be local to ColdFusion server or accessible on network  
|           | • If context = "nameservice": forward slash-delimited naming context for naming service. For example: Allaire//Doc/empobject. |
| **locale** | The name of the Java config that holds the properties file. For more information, see *Configuring and Administering ColdFusion*. |

**Usage**

In the **class** parameter, if "context=NameService", use a dot separator for the first part of the string. Use either of the following formats:

- "/Eng/CF"
- ".current/Eng.current/CF"

ColdFusion supports CORBA through the Dynamic Invocation Interface (DII). To use this function with CORBA objects, provide the name of the file that contains a string version of the IOR, or the object's naming context in the naming service. Provide the object's attributes, method names, and method signatures. This function supports user-defined types (structures, arrays, and sequences).

**Example**

```javascript
myobj = CreateObject("corba", "d:\temp\tester.ior", "ior", "visibroker") // uses IOR
myobj = CreateObject("corba", "/Eng/CF", "nameservice", "visibroker") // uses nameservice
myobj = CreateObject("corba", "d:\temp\tester.ior", "nameservice")
// uses nameservice and default configuration
```
CreateObject: Java or EJB object

Description

The CreateObject function can create a Java object, and, by extension, an EJB object.

Returns

A Java object.

Function syntax

```
CreateObject(type, class)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `type`    | Type of object to create.  
- com  
- corba  
- java  
- component  
- webservice  
  The default value of type is component. |
| `class`   | A Java class name |

Usage

Any Java class available in the class path that is specified in the ColdFusion Administrator can be loaded and used from ColdFusion with the CreateObject function. To access Java methods and fields:

1. Call the CreateObject function or the cfoject tag to load the class.
2. Use the `init` method, with appropriate arguments, to call an instance of the class. For example:

```
<cfset ret = myObj.init(arg1, arg2)>
```

Calling a public method on the object without first calling the "init" method invokes a static method. Arguments and return values can be any Java type (simple, array, object). If strings are passed as arguments, ColdFusion does the conversions; if strings are received as return values, ColdFusion does no conversion. Overloaded methods are supported if the number of arguments is different. Future enhancements will let you use cast functions that allow method signatures to be built more accurately.
**CreateObject: web service object**

**Description**

This function can create a web service object.

**Returns**

A web service object.

**Function syntax**

```
CreateObject(type, urltowsdl[, portname, wsdl2JavaArgs])

OR

CreateObject(type, urltowsdl, argStruct)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Type of object to create.</td>
</tr>
<tr>
<td></td>
<td>• com</td>
</tr>
<tr>
<td></td>
<td>• corba</td>
</tr>
<tr>
<td></td>
<td>• java</td>
</tr>
<tr>
<td></td>
<td>• component</td>
</tr>
<tr>
<td></td>
<td>• webservice</td>
</tr>
<tr>
<td></td>
<td>The default value of type is component.</td>
</tr>
<tr>
<td>urltowsdl</td>
<td>Specifies the URL to web service WSDL file. One of the following:</td>
</tr>
<tr>
<td></td>
<td>• The absolute URL of the web service</td>
</tr>
<tr>
<td></td>
<td>• The Name (string) assigned in the ColdFusion Administrator to the web service</td>
</tr>
<tr>
<td>portname</td>
<td>The port name for the web service. This value is case-sensitive and corresponds to the port element's name attribute under the service element. Specify this parameter if the web service contains multiple ports. If no port name is specified, ColdFusion uses the first port found in the WSDL.</td>
</tr>
</tbody>
</table>
### wsl2JavaArgs

A string containing a space-delimited list of arguments to pass to the WSDL2Java tool that generates Java stubs for the web services. Useful arguments include the following:

- `W` or `-noWrapped`: Turns off the special treatment of wrapped document/literal style operations.
- `a` or `-all`: Generates code for all elements in the WSDL, even unreferenced ones.
- `w` or `-wrapArrays`: Prefers building beans to straight arrays for wrapped XML array types. This switch is not included in the Axis documentation. For detailed information on valid arguments, see the Apache Axis WSDL2Java Reference.

### argStruct

A structure containing web service configuration arguments. For more information see Usage

### Usage

You can use the `CreateObject` function to create a web service. The `argStruct` structure can contain any combination of the following values:

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>Password set in the Administrator, if any</td>
<td>The password to use to access the web service. If the <code>webservice</code> attribute specifies a web service name configured in the Administrator, overrides any user name specified in the Administrator entry.</td>
</tr>
<tr>
<td>port</td>
<td></td>
<td>See <code>portname</code> in the Syntax Parameter table.</td>
</tr>
<tr>
<td>proxyPassword</td>
<td>http.proxyPassword system property, if any</td>
<td>The user’s password on the proxy server.</td>
</tr>
<tr>
<td>proxyPort</td>
<td>http.proxyPort system property, if any</td>
<td>The port to use on the proxy server.</td>
</tr>
<tr>
<td>proxyServer</td>
<td>http.proxyHost system property, if any</td>
<td>The proxy server required to access the webservice URL.</td>
</tr>
<tr>
<td>proxyUser</td>
<td>http.proxyUser system property, if any</td>
<td>The user ID to send to the proxy server.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| refreshWSDL   | no      | • yes: Reload the WSDL file and regenerate the artifacts used to consume the web service  
                 • no                                                |
| saveJava      | no      | • yes: Save the Java generated by the WSDL2Java converter that generates Java web service stubs. This code can be useful in debugging errors.  
                 • no                                                |
| timeout       | 0 (no time-out) | The time-out for retrieving the web service WSDL, in seconds. |
| username      | User name set in the Administrator, if any | The user name to use to access the web service. If the webservice attribute specifies a web service name configured in the Administrator, overrides any user name specified in the Administrator entry. |
| wsdl2javaArgs |         | See the Syntax parameter table.                                           |

Example

```cfs<cfscript>
    ws = CreateObject("webservice", 
        "http://www.xmethods.net/sd/2001/TemperatureService.wsdl");
    xlatstring = ws.getTemp(zipcode = "55987");
    writeoutput("The temperature at 55987 is " & xlatstring);
</cfscript>```
CreateODBCDate

Description

Creates an ODBC date object.

Returns

A date object, in normalized ODBC date format.

Category

Date and time functions

Function syntax

CreateODBCDate(date)

See also

CreateDate, CreateODBCDateTime

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date or date/time object in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

This function does not parse or validate values. To ensure that dates are entered and processed correctly (for example, to ensure that a day/month/year entry is not confused with a month/day/year entry, and so on), Adobe recommends that you parse entered dates with the DateFormat function, using the mm-dd-yyyy mask, into three elements. Ensure that values are within appropriate ranges; for example, to validate a month value, use the attributes validate = "integer" and range = "1,12".

Example
<h3>CreateODBCDate Example</h3>

<cfif IsDefined("form.year")>
<p>Your date value, generated with CreateDateTime:</p>
<cfset yourDate = CreateDateTime(form.year, form.month, form.day, form.hour, form.minute, form.second)>
<cfoutput>
<ul>
  <li>Formatted with CreateDate: #CreateDate(form.year, form.month, form.day)#</li>
  <li>Formatted with CreateDateTime: #CreateDateTime(form.year, form.month, form.day, form.hour, form.minute, form.second)#</li>
  <li>Formatted with CreateODBCDate: #CreateODBCDate(yourDate)#</li>
  <li>Formatted with CreateODBCDateTime: #CreateODBCDateTime(yourDate)#</li>
</ul>
<p>The same value can be formatted with DateFormat:</p>
<ul>
  <li>Formatted with CreateDate and DateFormat: #DateFormat(CreateDate(form.year,form.month, form.day), "mmm-dd-yyyy")#</li>
  <li>Formatted with CreateDateTime and DateFormat: #DateFormat(CreateDateTime(form.year, form.month, form.day, form.hour, form.minute, form.second))#</li>
  <li>Formatted with CreateODBCDate and DateFormat: #DateFormat(CreateODBCDate(yourDate), "mmm d, yyyy")#</li>
  <li>Formatted with CreateODBCDateTime and DateFormat: #DateFormat(CreateODBCDateTime(yourDate), "d/m/yy")#</li>
</ul>
</cfoutput>
</cfif>
</cfif>

<cfform action="createodbcdate.cfm" method="POST">
<p>Enter the year, month and day, as integers:
<pre>
Year  <cfinput type="Text" name="year" value="1998" validate="integer" required="Yes">
Month  <cfinput type="Text" name="month" value="6" range="1,12" message="Please enter a month (1-12)" validate="integer" required="Yes">
Day  <cfinput type="Text" name="day" value="8" range="1,31" message="Please enter a day of the month (1-31)" validate="integer" required="Yes">
Hour  <cfinput type="Text" NAME="hour" value="16" range="0,23" message="You must enter an hour (0-23)" validate="integer" required="Yes">
Minute  <cfinput type="Text" name="minute" value="12" range="0,59" message="You must enter a minute value (0-59)" validate="integer" required="Yes">
Second  <cfinput type="Text" name="second" value="0" range="0,59" message="You must enter a value for seconds (0-59)" validate="integer" required="Yes">
</pre>
</cfform>
CreateODBCDateTime

Description

Creates an ODBC date-time object.

Returns

A date-time object, in ODBC timestamp format.

Category

Date and time functions

Function syntax

CreateODBCDateTime(date)

See also

CreateDateTime, CreateODBCDate, CreateODBCTime, Now; Evaluation and type conversion issues in Data type conversion in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date-time object in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

When passing a date-time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date-time object.

Example
<!--- This example shows how to use CreateDate, CreateDateTime, CreateODBCDate, and CreateODBCDateTime Example --->

<h3>CreateODBCDateTime Example</h3>

<cfif IsDefined("form.year")>

Your date value, generated using CreateDateTime:

<cfset yourDate = CreateDateTime (form.year, form.month, form.day,
form.hour,form.minute, form.second)>

<cfoutput>
<ul>
<li>Formatted with CreateDate: #CreateDate(form.year, form.month,form.day)#
<li>Formatted with CreateDateTime: #CreateDateTime(form.year,form.month,
form.day,form.hour,form.minute,form.second)#
<li>Formatted with CreateODBCDate: #CreateODBCDate(yourDate)#
<li>Formatted with CreateODBCDateTime: #CreateODBCDateTime(yourDate)#
</ul>
<p>The same value can be formatted with DateFormat:
<ul>
<li>Formatted with CreateDate and DateFormat: #DateFormat(CreateDate(form.year,form.month,form.day), "mmm-dd-yyyy")#
<li>Formatted with CreateDateTime and DateFormat: #DateFormat(CreateDateTime(form.year,form.month,form.day,
form.hour,form.minute,form.second))#
<li>Formatted with CreateODBCDate and DateFormat: #DateFormat(CreateODBCDate(yourDate), "mmmm d, yyyy")#
<li>Formatted with CreateODBCDateTime and DateFormat: #DateFormat(CreateODBCDateTime(yourDate), "d/m/yy")#
</ul>
</cfoutput>
</cfif>

<cfform ACTION="createodbcdatetime.cfm" METHOD="POST">

Enter a year, month and day, as integers:
<pre>
Year  <cfinput TYPE="Text" NAME="year" VALUE="1998" VALIDATE="integer" REQUIRED="Yes">
Month  <cfinput TYPE="Text" NAME="month"VALUE="6" RANGE="1,12"
MESSAGE="Enter a month (1-12)" VALIDATE="integer" REQUIRED="Yes">
Day  <cfinput TYPE="Text" NAME="day" VALUE="8" RANGE="1,31"
MESSAGE="Enter a day of the month (1-31)" VALIDATE="integer" REQUIRED="Yes">
Hour  <cfinput TYPE="Text" NAME="hour" VALUE="16" RANGE="0,23"
MESSAGE="You must enter an hour (0-23)" VALIDATE="integer" REQUIRED="Yes">
Minute  <cfinput TYPE="Text" NAME="minute" VALUE="12" RANGE="0,59"
MESSAGE="You must enter a minute value (0-59)" VALIDATE="integer" REQUIRED="Yes">
Second  <cfinput TYPE="Text" NAME="second" VALUE="0" RANGE="0,59"
MESSAGE="You must enter a seconds value (0-59)" VALIDATE="integer"
REQUIRED="Yes">
</pre>
<p><input TYPE="Submit" NAME=""> <input TYPE="RESET">
</cfform>
CreateODBCTime

Description

Creates an ODBC time object.

Returns

A time object, in ODBC timestamp format.

Category

Date and time functions

Function syntax

CreateODBCTime(date)

See also

CreateODBCDateTime, CreateTime, Evaluation and type conversion issues in Data type conversion in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

When passing a date-time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date-time object.

Example
<h3>CreateODBCTime Example</h3>

<CFIF IsDefined("form.hour")>
Your time value, created with CreateTime...
<CFSET yourTime = CreateTime(form.hour, form.minute, form.second)>
(cfoutput>
<ul>
<li>Formatted with CreateODBCTime: #CreateODBCTime(yourTime)#
<li>Formatted with TimeFormat: #TimeFormat(yourTime)#
</ul></cfoutput>
</CFIF>

<CFORM action="createodbcstime.cfm" METHOD="post">
<pre>
Hour  <CFINPUT TYPE="Text" NAME="hour" VALUE="16" RANGE="0,23" MESSAGE="You must enter an hour (0-23)" VALIDATE="integer" REQUIRED="Yes">
Minute <CFINPUT TYPE="Text" NAME="minute" VALUE="12" RANGE="0,59" MESSAGE="You must enter a minute value (0-59)" VALIDATE="integer" REQUIRED="Yes">
Second <CFINPUT TYPE="Text" NAME="second" VALUE="0" RANGE="0,59" MESSAGE="You must enter a value for seconds (0-59)" VALIDATE="integer" REQUIRED="Yes">
</pre>
<p><INPUT TYPE="Submit" NAME=""> <INPUT TYPE="RESET">
</cform>
CreateTime

Description

Creates a time variable.

Returns

A time variable.

Category

Date and time functions

Function syntax

CreateTime(hour, minute, second)

See also

CreateTime, CreateDateTime; Evaluation and type conversion issues in Data type conversion in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hour</td>
<td>Number in the range 0-23</td>
</tr>
<tr>
<td>minute</td>
<td>Number in the range 0-59</td>
</tr>
<tr>
<td>second</td>
<td>Number in the range 0-59</td>
</tr>
</tbody>
</table>

Usage

CreateTime is a subset of CreateDateTime. A time variable is a special case of a date-time variable. The date part of a time variable is set to December 30, 1899.

Example
CreateTime Example

<cfif IsDefined("FORM.hour")>
Your time value, presented using CreateTime time function:

<cfset yourTime = CreateTime(FORM.hour, FORM.minute, FORM.second)>

<li>Formatted with timeFormat: #TimeFormat(yourTime)#</li>
<li>Formatted with timeFormat and hh:mm:ss: #TimeFormat(yourTime, 'hh:mm:ss')#</li>
</cfoutput>
</cfif>

<cfform action="createtime.cfm" method="post">
<pre>
Hour &nbsp;&nbsp;&nbsp;<cfinput type="text" name="hour" value="16" range="0,23">
Message="You must enter an hour (0-23)" validate="integer" required="Yes">
Minute <cfinput type="text" name="minute" value="12" range="0,59">
Message="You must enter a minute value (0-59)" validate="integer"
required="Yes">
Second <cfinput type="text" name="second" value="0" range="0,59">
Message="You must enter a value for seconds (0-59)" validate="integer"
required="Yes">
</pre>
</cfform>
CreateTimeSpan

Description

Creates a date/time object that defines a time period. You can add or subtract it from other date-time objects and use it with the cachedWithin attribute of cfquery.

Returns

A date-time object. (A Java Double)

Category

Date and time functions

Function syntax

CreateTimeSpan(days, hours, minutes, seconds)

See also

CreateDateDateTime, DateAdd, DateConvert; Defining application-level settings and variables in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>Integer in the range 0-32768; number of days in time period</td>
</tr>
<tr>
<td>hours</td>
<td>Number of hours in time period</td>
</tr>
<tr>
<td>minutes</td>
<td>Number of minutes in time period</td>
</tr>
<tr>
<td>seconds</td>
<td>Number of seconds in time period</td>
</tr>
</tbody>
</table>

Usage

Creates a special date-time object that should be used only to add and subtract from other date-time objects or with the cfquery cachedWithin attribute. If you use the cachedWithin attribute of cfquery, and the original query date falls within the time span you define, cached query data is used. In this case, the CreateTimeSpan function is used to define a period of time from the present backwards. The cachedWithin attribute takes effect only if you enable query caching in the ColdFusion Administrator. For more information, see cfquery.

Example
<!--- This example shows the use of CreateTimeSpan with cfquery --->
<h3>CreateTimeSpan Example</h3>

<!--- define startrow and maxrows to facilitate 'next N' style browsing --->
<CFPARAM name = "MaxRows" default = "10">
<CFPARAM name = "StartRow" default = "1">

<!--- Query database for information, if cached database information has not been updated in the last six hours. -------->
<cfoutput>
<cfquery name = "GetParks" datasource = "cfdocexamples"
cachedWithin = "#CreateTimeSpan(0, 6, 0, 0)#">
SELECT PARKNAME, REGION, STATE
FROM Parks
ORDER by ParkName, State
</cfquery>
</cfoutput>

<!--- build HTML table to display query --->
<TABLE cellpadding = 1 cellspacing = 1>
<tr>
<TD colspan = 2 bgcolor = f0f0f0>
<B><I>Park Name</I></B>
</TD>
<TD bgcolor = f0f0f0>
<B><i>Region</I></B>
</TD>
<TD bgcolor = f0f0f0>
<B><I>State</I></B>
</TD>
</tr>
<!--- Output query, define startrow and maxrows. Use query variable CurrentCount to track the row you are displaying. --->
<cfoutput query = "GetParks" StartRow = "#StartRow#"
MAXROWS = "#MaxRows#">
<tr>
<TD valign = top bgcolor = ffffed>
<B>#GetParks.CurrentRow#</B>
</TD>
<TD valign = top>
<FONT SIZE = "-1">#ParkName#</FONT>
</TD>
<TD valign = top>
<FONT SIZE = "-1">#Region#</FONT>
</TD>
<TD valign = top>
<FONT SIZE = "-1">#State#</FONT>
</TD>
</tr>
</cfoutput>
</table>

<!--- If number of records is less than or equal to number of rows, offer link to same page, with startrow value incremented by maxrows (in this example, incremented by 10). --->
<cfif (StartRow + MaxRows) LTE GetParks.RecordCount>
<a href = "cfquery.cfm?startrow = <cfoutput>#StartRow + MaxRows#</cfoutput>">See next <cfoutput>#MaxRows#</cfoutput> rows</A>
</cfif>
CreateUUID

Description

Creates a Universally Unique Identifier (UUID). A UUID is a 35-character string representation of a unique 128-bit integer.

Returns

A ColdFusion format UUID, in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx, where x is a hexadecimal digit (0-9 or A-F). (The character groups are 8-4-4-16.)

Category

Other functions

Function syntax

CreateUUID()

Usage

The ColdFusion UUID generation algorithm uses the unique time-of-day value, the IEEE 802 Host ID, and a cryptographically strong random number generator to generate UUIDs that conform to the principles laid out in the draft IEEE RFC "UUIDs and GUIDs." The ColdFusion UUID format is as follows: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx (8-4-4-16). This does not conform to the Microsoft/DCE standard, which is as follows: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx (8-4-4-12). There are UUID test tools and a user-defined function called CreateGUID, which converts CFML UUIDs to UUID/Microsoft GUID format, available on the web at www.cflib.org. Use this function to generate a persistent identifier in a distributed environment. To a very high degree of certainty, this function returns a unique value; no other invocation on the same or any other system returns the same value. UUIDs are used by distributed computing frameworks, such as DCE/RPC, COM+, and CORBA. In ColdFusion, you can use UUIDs as primary table keys for applications in which data is stored in shared databases. In such cases, using numeric keys can cause primary-key constraint violations during table merges. Using UUIDs, you can eliminate these violations.

Example

```
<h3>CreateUUID Example</h3>
<p>This example uses CreateUUID to generate a UUID when you submit the form. You can submit the form more than once. </p>
<!--- Checks whether the form was submitted; if so, creates UUID. --->
<cfif IsDefined("Form.CreateUUID") Is True>
    <hr>
    <p>Your new UUID is: <cfoutput>#CreateUUID()#</cfoutput></p>
</cfif>
<form action = "createuuid.cfm">
    <p><input type = "Submit" name = "CreateUUID"></p>
</form>
```
CSRFGenerateToken

Description

Provides a random token and stores it in the session. You can also provide a specific key to store in the session.

Returns

Token

Category

Display and formatting functions

Syntax

CSRFGenerateToken([key] [,forceNew])

See also

CSRFVerifyToken

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>optional</td>
<td>A random token is generated based on the key provided. This key is stored in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the session.</td>
</tr>
<tr>
<td>forceNew</td>
<td>optional</td>
<td>If set to true, a new token is generated every time the method is called.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If false, in case a token exists for the key, the same key is returned.</td>
</tr>
</tbody>
</table>

Usage

Use this function to create a random token and store it in the session.

Example

```cftext
<cfset csrfToken=CSRFGenerateToken() />
<cfform method="post" action="sayHello.cfm">
    <cfinput name="userName" type="text" />
    <cfinput name="token" value="#csrfToken#" type="hidden" />
    <cfinput name="submit" value="Say Hello!!" type="submit" />
</cfform>
```
CSRFVerifyToken

Description

Validates the given token against the same stored in the session for a specific key.

Returns

a Boolean value. true is successful.

Category

Display and formatting functions

Syntax

CSRFVerifyToken(token [,key])

See also

CSRFGenerateToken

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>required</td>
<td></td>
<td>Token that to be validated against the token stored in the session.</td>
</tr>
<tr>
<td>key</td>
<td>Optional</td>
<td></td>
<td>The key against which the token be searched.</td>
</tr>
</tbody>
</table>

Usage

Use this function to validate the given token against the same stored in the session for a specific key.

Example

```cfc
cfset token=form.token>
cfset validate = CSRFverifyToken(token)>
cfoutput &#validate#</cfoutput>
```
DateAdd

Description

Adds units of time to a date.

Returns

A date/time object.

Category

Date and time functions

Function syntax

```
DateAdd("datepart", number, "date")
```

See also

DateConvert, DatePart, CreateTimeSpan

History

ColdFusion MX 6.1: Added the datepart character L or l to represent milliseconds.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datepart</td>
<td>String:</td>
</tr>
<tr>
<td></td>
<td>• yyyy: Year</td>
</tr>
<tr>
<td></td>
<td>• q: Quarter</td>
</tr>
<tr>
<td></td>
<td>• m: Month</td>
</tr>
<tr>
<td></td>
<td>• y: Day of year</td>
</tr>
<tr>
<td></td>
<td>• d: Day</td>
</tr>
<tr>
<td></td>
<td>• w: Weekday</td>
</tr>
<tr>
<td></td>
<td>• ww: Week</td>
</tr>
<tr>
<td></td>
<td>• h: Hour</td>
</tr>
<tr>
<td></td>
<td>• n: Minute</td>
</tr>
<tr>
<td></td>
<td>• s: Second</td>
</tr>
<tr>
<td></td>
<td>• l: Millisecond</td>
</tr>
</tbody>
</table>

| number    | Number of units of datepart to add to date (positive, to get dates in the future; negative, to get dates in the past). Number must be an integer. |
| date      | Date/time object, in the range 100 AD-9999 AD. |
The *datepart* specifiers *y*, *d*, and *w* add a number of days to a date. When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

**Example**

```cfc
<cfset date="(ts '2433-09-01 23:59:59')">  
<cfoutput>#date#</cfoutput>  
<cfset diff=30>  
<cfset posdateresult=DateAdd("s",diff,date1)>  
<cfoutput>#posdateresult#</cfoutput>
```

**Example**

```cfc
<cfparam name="value" default="70">  
<cfparam name="type" default="m">  

<!---- If numbers passed, then use those. --->
<cfif IsDefined("form.value")]  
<cfset value = form.value>  
</cfif>  
<cfif IsDefined("form.type")]  
<cfset type = form.type>  
</cfif>  

<cquery name="GetMessages" datasource="cfdocexamples">  
SELECT UserName, Subject, Posted  
FROM Messages  
</cquery>  

<p>This example uses DateAdd to determine when a message in the database will expire. Currently, messages older than <cfoutput>#value#</cfoutput> are expired.

```cfc
<cfswitch expression="#type#">  
<cfcase value="yyyy">years</cfcase>  
<cfcase value="q">quarters</cfcase>  
<cfcase value="m">months</cfcase>  
<cfcase value="y">days of year</cfcase>  
<cfcase value="w">weekdays</cfcase>  
<cfcase value="ww">weeks</cfcase>  
<cfcase value="h">hours</cfcase>  
<cfcase value="n">minutes</cfcase>  
<cfcase value="s">seconds</cfcase>  
<cfdefaultcase>years</cfdefaultcase>  
</cfswitch>  
are expired.
```

```cfc
<table>  
<tr>  
<td>UserName</td>  
<td>Subject</td>  
<td>Posted</td>  
</tr>
```
<cfoutput query="GetMessages">
<tr>
<td>#UserName#</td>
<td>#Subject#</td>
<td>#Posted#<cfif DateAdd(type, value, posted) LT Now()><font color="red">EXPIRED</font></cfif></td>
</tr>
</cfoutput>
</table>

<cfform action="#CGI.Script_Name#" method="post">
Select an expiration value:
<input type="Text" name="value" value="#value#" message="Please enter whole numbers only" validate="integer" required="Yes">
<select name="type">
<option value="yyyy">years</option>
<option value="m" selected>months</option>
<option value="d">days</option>
<option value="ww">weeks</option>
<option value="h">hours</option>
<option value="n">minutes</option>
<option value="s">seconds</option>
</select>
</cfform>
<input type="Submit" value="Submit">
</cfform>
**DateCompare**

**Description**
Performs a full date/time comparison of two dates.

**Returns**
- -1, if `date1` is earlier than `date2`
- 0, if `date1` is equal to `date2`
- 1, if `date1` is later than `date2`

**Category**
*Date and time functions*

**Function syntax**

```
DateCompare("date1", "date2" [, "datePart"])
```

**See also**
`CreateDateTime`, `DatePart`

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>date1</code></td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
<tr>
<td><code>date2</code></td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
<tr>
<td><code>datePart</code></td>
<td>Optional. String. Precision of the comparison.</td>
</tr>
<tr>
<td></td>
<td>• s-Precise to the second (default)</td>
</tr>
<tr>
<td></td>
<td>• n-Precise to the minute</td>
</tr>
<tr>
<td></td>
<td>• h-Precise to the hour</td>
</tr>
<tr>
<td></td>
<td>• d-Precise to the day</td>
</tr>
<tr>
<td></td>
<td>• m-Precise to the month</td>
</tr>
<tr>
<td></td>
<td>• yyyy-Precise to the year</td>
</tr>
</tbody>
</table>

**Usage**

When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

**Example**

```
<h3>DateCompare Example</h3>
<p>The DateCompare function compares two date/time values.
<cfif IsDefined("FORM.date1")>
  <cfif IsDate(FORM.date1) and IsDate(FORM.date2)>
    <cfset comparison = DateCompare(FORM.date1, FORM.date2, FORM.precision)>
```
<!--- Switch on the variable to give various responses. --->
<cfswitch expression = #comparison#>
  <cfcase value = "-1">
    <h3>\#DateFormat(FORM.date1)\ #TimeFormat(FORM.date1)\ (Date 1) is earlier than \#DateFormat(FORM.date2)\ #TimeFormat(FORM.date2)\ (Date 2)\</h3>
    <i>The dates are not equal</i>
  </cfcase>
  <cfcase value = "0">
    <h3>\#DateFormat(FORM.date1)\ #TimeFormat(FORM.date1)\ (Date 1) is equal to \#DateFormat(FORM.date2)\ #TimeFormat(FORM.date2)\ (Date 2)\</h3>
    <i>The dates are equal!</i>
  </cfcase>
  <cfcase value = "1">
    <h3>\#DateFormat(FORM.date1)\ #TimeFormat(FORM.date1)\ (Date 1) is later than \#DateFormat(FORM.date2)\ #TimeFormat(FORM.date2)\ (Date 2)\</h3>
    <i>The dates are not equal</i>
  </cfcase>
  <CFDEFAULTCASE>
    <h3>This is the default case</h3>
  </CFDEFAULTCASE>
</cfswitch>
<cfelse>
  <h3>Enter two valid date values</h3>
</cfif>
</cfif>
<form action = "datecompare.cfm" method="post">
  <hr size = "2" color = "#0000A0">
  <p>Date 1
    <br><input type = "Text" name = "date1" value = "\#DateFormat(Now())\ #TimeFormat(Now())\">
  </p>
  <p>Date 2
    <br><input type = "Text" name = "date2" value = "\#DateFormat(Now())\ #TimeFormat(Now())\">
  </p>
  <p>Specify precision to the:
    <br><select name = "precision">
      <option value = "s">Second</option>
      <option value = "n">Minute</option>
      <option value = "h">Hour</option>
      <option value = "d">Day</option>
      <option value = "m">Month</option>
    </select>
  </p>
</form>
<option value="yyyy">
Year
</option>
</select>
<p><input type="Submit" value="Compare these dates" name=""/>

<input type = "reset">
</form>
**DateConvert**

**Description**

Converts local time to Coordinated Universal Time (UTC), or UTC to local time. The function uses the daylight savings settings in the executing computer to compute daylight savings time, if necessary.

**Returns**

UTC- or local-formatted time object.

**Category**

Date and time functions

**Function syntax**

```
DateConvert("conversion-type", "date")
```

**See also**

GetTimeZoneInfo, CreateDateTime, DatePart

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conversion-type</td>
<td>• local2Utc: Converts local time to UTC time.</td>
</tr>
<tr>
<td></td>
<td>• utc2Local: Converts UTC time to local time.</td>
</tr>
<tr>
<td>date</td>
<td>Date and time string or a variable that contains one. To create, use CreateDateTime.</td>
</tr>
</tbody>
</table>

**Usage**

When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

**Note**

You can pass the CreateDate or Now function as the date parameter of this function; for example: #DateConvert(CreateDate(2007, 4, 10))#

**Example**

```
<h3>DateConvert Example</h3>
<!---- This shows conversion of current date - time to UTC and back. --->
<cfset curDate = Now()>
<p><cfoutput>The current date and time: #curDate#. </cfoutput></p>
<cfset utcDate = DateConvert("local2utc", curDate)>
<cfoutput>
```
<p>The current date and time converted to UTC time: #utcDate#.</p>
</cfoutput>

<!-- This code checks whether form was submitted. If so, the code generates the CFML date with the CreateDateTime function. -->
<cfif IsDefined("FORM.submit")>
<cfset yourDate = CreateDateTime(FORM.year, FORM.month, FORM.day,
FORM.hour,FORM.minute, FORM.second)>
<p><cfoutput>Your date value, presented as a ColdFusion date/time string:#yourdate#.</cfoutput></p>
<cfset yourUTC = DateConvert("local2utc", yourDate)>
<p><cfoutput>Your date and time value, converted to Coordinated Universal Time (UTC): #yourUTC#. </cfoutput></p>
<p><cfoutput>Your UTC date and time, converted back to local date and time: #DateConvert("utc2local", yourUTC)#.</cfoutput></p>
</cfif>
<cfelse>
Type the date and time, and press Enter to see the conversion.
</cfif>

<Hr size = "2" color = "#0000A0">
<form action = "dateconvert.cfm" method="post">
<p>Enter year, month and day in integer format for date value to view:
<table cellspacing = "2" cellpadding = "2" border = "0">
<tr>
<td>Year</td>
<td><input type = "Text" name = "year" value = "1998" validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Month</td>
<td><input type = "Text" name = "month" value = "6" range = "1,12" message = "Enter a month (1-12)" validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Day</td>
<td><input type = "Text" name = "day" value = "8" range = "1,31" message = "Enter a day of the month (1-31)" validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Hour</td>
<td><input type = "Text" name = "hour" value = "16" range = "0,23" message = "You must enter an hour (0-23)" validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Minute</td>
<td><input type = "Text" name = "minute" value = "12" range = "0,59" message = "You must enter a minute value (0-59)" validate = "integer" required = "Yes"></td></tr>
<tr>
<td>Second</td>
<td><input type = "Text" name = "second" value = "0" range = "0,59" message = "You must enter a value for seconds (0-59)" validate = "integer" required = "Yes"></td></tr>
<tr>
<td><input type = "Submit" name = "submit" value = "Submit"></td>
</table>
</p>
</form>
```html
<td><input type = "RESET"></td></tr>
</table>
```
DateDiff

Description

Determines the integer number of units by which \textit{date1} is less than \textit{date2}.

Returns

A number of units, of type \textit{datepart}.

Category

Date and time functions

Function syntax

\begin{verbatim}
DateDiff("datepart", "date1", "date2")
\end{verbatim}

See also

\textit{DateAdd}, \textit{DatePart}, \textit{CreateTimeSpan}

History

ColdFusion MX:

- Changed how negative date differences are calculated: this function calculates negative date differences correctly; its output may be different from that in earlier releases.
- Changed the \textit{w} and \textit{ww} masks; they determine the number of full weeks between the two dates.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| datepart  | String that specifies the units in which to count; for example yyyy requests a date difference in whole years.  
- yyyy: Years  
- q: Quarters  
- m: Months  
- y: Days of year (same as d)  
- d: Days  
- w: Weekdays (same as ww)  
- ww: Weeks  
- h: Hours  
- n: Minutes  
- s: Seconds |
| date1     | Date/time object, in the range 100 AD-9999 AD. |
| date2     | Date/time object, in the range 100 AD-9999 AD. |
Usage

The DateDiff function determines the number of complete datepart units between the two dates; for example, if the datepart parameter is "m" and the dates differ by 55 days, the function returns 1. Enclose string constant dates in quotation marks. If the text contains only numbers (such as 1932), and is not surrounded by quotation marks, ColdFusion interprets it as a date/time object, resulting in an incorrect value.

Example

```cfc
<cfif IsDefined("form.value")>
  <cfset value = form.value>
</cfif>
<cfif IsDefined("form.type")>
  <cfset type = form.type>
</cfif>

<cfif IsDefined("form.date1") and IsDefined("form.date2")>
  <cfif IsDate(form.date1) and IsDate(form.date2)>
    <p>This example uses DateDiff to determine the difference in dateparts between date1 and date2.
    <cfswitch expression = "#form.type#">
      <cfcase value="yyyy">years</cfcase>
      <cfcase value="q">quarters</cfcase>
      <cfcase value="m">months</cfcase>
      <cfcase value="y">days</cfcase>
      <cfcase value="d">days</cfcase>
      <cfcase value="w">weekdays</cfcase>
      <cfcase value="ww">weeks</cfcase>
      <cfcase value="h">hours</cfcase>
      <cfcase value="n">minutes</cfcase>
      <cfcase value="s">seconds</cfcase>
      <cfdefaultcase>years</cfdefaultcase>
    </cfswitch>
    <p>The difference is <cfoutput>#Abs(DateDiff(type, form.date2, form.date1))#</cfoutput>.
  </cfif>
<cfelse>
  <p>The two dates are equal! Try changing one of the values ...</p>
</cfif>
```
<cfelse>
<p>Please enter two valid date/time values, formatted like this:
<cfoutput>#DateFormat(Now())#</cfoutput></p></cfif>
</cfif>
<form action="index.cfm" method="post">
<pre>
Date 1
<input type="Text" name="date1" value="<cfoutput>#DateFormat(Now())#</cfoutput>">
Date 2
<input type="Text" name="date2" value="<cfoutput>#DateFormat(Now())#</cfoutput>">
What kind of unit to show difference?
<select name="type">
<option value="yyyy" selected>years
<option value="q">quarters
<option value="m">months
<option value="y">days of year
<option value="d">days
<option value="w">weekdays
<option value="ww">weeks
<option value="h">hours
<option value="n">minutes
<option value="s">seconds
</select>
</pre>
</form>
<input type="Submit" name=""/><input type="Reset"/>
</form>
DateFormat

Description

Formats a date value using U.S. date formats. For international date support, use LSDateFormat.

Returns

A text string representing the date formatted according to the mask. If no mask is specified, returns the value in dd-mmm-yy format.

Category

Date and time functions

Function syntax

```
DateFormat("date" \[, "mask" ])
```

See also

Now, CreateDate, LSDateFormat, LSParseDateTime, LSTimeFormat, TimeFormat, ParseDateTime

History

ColdFusion MX: Added support for the following mask parameter options: short, medium, long, and full.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>
Characters that show how ColdFusion displays a date:

- **d**: Day of the month as digits; no leading zero for single-digit days.
- **dd**: Day of the month as digits; leading zero for single-digit days.
- **ddd**: Day of the week as a three-letter abbreviation.
- **dddd**: Day of the week as its full name.
- **m**: Month as digits; no leading zero for single-digit months.
- **mm**: Month as digits; leading zero for single-digit months.
- **mmm**: Month as a three-letter abbreviation.
- **mmmm**: Month as its full name.
- **yy**: Year as last two digits; leading zero for years less than 10.
- **yyyy**: Year represented by four digits.
- **gg**: Period/era string. Ignored. Reserved.

The following masks tell how to format the full date and cannot be combined with other masks:

- **short**: equivalent to m/d/y
- **medium**: equivalent to mmm d, yyyy
- **long**: equivalent to mmmm d, yyyy
- **full**: equivalent to dddd, mmmm d, yyyy

### Usage

When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

**Note**

You can pass the `CreateDate` function or `Now` function as the date parameter of this function; for example: `#DateFormat(CreateDate(2001, 3, 3))#`.

Date and time values in database query results can vary in sequence and formatting unless you use functions to format them. To ensure that application users correctly understand displayed dates and times, Adobe recommends that you use this function and the `LSDateFormat`, `TimeFormat`, and `LSTimeFormat` functions to format resultset values. For more information and examples, see TechNote, "ColdFusion Server (5 and 4.5.x) with Oracle: Formatting Date and Time Query Results," on the website at [go.adobe.com/kb/ts_tn_18070_en-us](go.adobe.com/kb/ts_tn_18070_en-us).

**Note**

The `DateFormat` function is best used for formatting output, not for formatting input. For formatting input, use one of the date/time creation functions (for example, `CreateDate`) instead.

### Example
<cfset todayDate = Now()>
<body>
<h3>Date Format Example</h3>
<p>Today's date is <cfoutput>#todayDate#</cfoutput>.</p>
<p>Using DateFormat, we can display that date in different ways:</p>
<cfoutput>
<ul>
<li>#DateFormat(todayDate)#
<li>#DateFormat(todayDate, "mmm-dd-yyyy")#
<li>#DateFormat(todayDate, "mmmm d, yyyy")#
<li>#DateFormat(todayDate, "mm/dd/yyyy")#
<li>#DateFormat(todayDate, "d-mmm-yyyy")#
<li>#DateFormat(todayDate, "ddd, mmmm dd, yyyy")#
<li>#DateFormat(todayDate, "short")#
<li>#DateFormat(todayDate, "medium")#
<li>#DateFormat(todayDate, "long")#
<li>#DateFormat(todayDate, "full")#
</ul>
</cfoutput>
DateTimeFormat

Description

Formats date and time values using date and time formatting conventions.

Returns

A formatted date and time value.

History

ColdFusion 10: Added this function.

Syntax

dateTimeFormat (date)
dateTimeFormat (_date [, mask])
dateTimeFormat (_date [, mask, _timeZone])

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Required. A date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>
Optional. Characters that show how ColdFusion displays a date:

- **d**: Day of the month as digits; no leading zero for single-digit days.
- **dd**: Day of the month as digits; leading zero for single-digit days.
- **ddd**: Day of the week as a three-letter abbreviation.
- **dddd**: Day of the week as its full name.
- **m**: Month as digits; no leading zero for single-digit months.
- **mm**: Month as digits; leading zero for single-digit months.
- **mmm**: Month as a three-letter abbreviation.
- **mmmm**: Month as its full name.
- **yy**: Year as last two digits; leading zero for years less than 10.
- **yyyy**: Year represented by four digits.
- **gg**: Period/era string.
- **h**: Hours; no leading zero for single-digit hours (12-hour clock)
- **hh**: Hours; leading zero for single-digit hours (12-hour clock)
- **H**: Hours; no leading zero for single-digit hours (24-hour clock)
- **HH**: Hours; leading zero for single-digit hours (24-hour clock)
- **n**: Minutes; no leading zero for single-digit minutes
- **nn**: Minutes; a leading zero for single-digit minutes
- **s**: Seconds; no leading zero for single-digit seconds
- **ss**: Seconds; leading zero for single-digit seconds
- **l** or **L**: Milliseconds, with no leading zeros
- **t**: One-character time marker string, such as A or P
- **tt**: Multiple-character time marker string, such as AM or PM

The following masks tell how to format the full date and time and cannot be combined with other masks:

- **short**: Equivalent to **m/d/y h:mm tt**
- **medium**: Equivalent to **mmm d, yyyy h:mm:ss tt**
- **long**: Medium followed by three-letter time zone; as in, **mmmm d, yyyy h:mm:ss tt EST**
- **full**: Equivalent to **dddd, mmmm d, yyyy h:mm:ss tt EST**

The function also follows Java date time mask. For details, see the section Date and Time Patterns on this page.
timeZone

The time-zone information. You can specify in either of the following formats:

- Abbreviation such as GMT or PST
- Full name such as Europe/Dublin

By default, this is the time-zone followed by the system.

Example

```
<cfset todayDateTime = Now()>
<body>
<h3>DateTimeFormat Example</h3>
<p>Today's date and time are <cfoutput>#todayDateTime#</cfoutput>.</p>
<p>Using DateTimeFormat, we can display that date and time in different ways:<br>
<cfoutput>
<ul>
  <li>#DateTimeFormat(todayDateTime)#
  <li>#DateTimeFormat(todayDateTime, "yyyy.MM.dd G 'at' HH:nn:ss z")#
  <li>#DateTimeFormat(todayDateTime, "EEE, MMM d, 'yy")#
  <li>#DateTimeFormat(todayDateTime, "h:nn a")#
  <li>#DateTimeFormat(todayDateTime, "hh 'o'clock' a, zzzz")#
  <li>#DateTimeFormat(todayDateTime, "K:nn a, z")#
  <li>#DateTimeFormat(todayDateTime, "yyyyy.MMMMM.dd GGG hh:nn aaa")#
  <li>#DateTimeFormat(todayDateTime, "EEE, d MMM yyyy HH:nn:ss Z")#
  <li>#DateTimeFormat(todayDateTime, "yyMMddHHnnsZ", "GMT")#
</ul>
</cfoutput>
```

**DatePart**

**Description**

Extracts a part from a date value.

**Returns**

Part of a date, as an integer.

**Category**

*Date and time functions*

**Function syntax**

```
DatePart("datepart", "date")
```

**See also**

*DateAdd*, *DateConvert*

**History**

ColdFusion MX 6.1: Added the *datepart* character L or l to represent milliseconds.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datepart</td>
<td>String:</td>
</tr>
<tr>
<td></td>
<td>• yyyy: Year</td>
</tr>
<tr>
<td></td>
<td>• q: Quarter</td>
</tr>
<tr>
<td></td>
<td>• m: Month</td>
</tr>
<tr>
<td></td>
<td>• y: Day of year</td>
</tr>
<tr>
<td></td>
<td>• d: Day</td>
</tr>
<tr>
<td></td>
<td>• w: Weekday</td>
</tr>
<tr>
<td></td>
<td>• ww: Week</td>
</tr>
<tr>
<td></td>
<td>• h: Hour</td>
</tr>
<tr>
<td></td>
<td>• n: Minute</td>
</tr>
<tr>
<td></td>
<td>• s: Second</td>
</tr>
<tr>
<td></td>
<td>• l: Millisecond</td>
</tr>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

**Usage**

When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

**Example**
<cffset todayDate = Now()>
<h3>DatePart Example</h3>
<p>Today's date is <cfoutput>#todayDate#</cfoutput>. Using datepart, we extract an integer representing the dateparts from that value</p>
<cfoutput>
<ul>
<li>year: #DatePart("yyyy", todayDate)#</li>
<li>quarter: #DatePart("q", todayDate)#</li>
<li>month: #DatePart("m", todayDate)#</li>
<li>day of year: #DatePart("y", todayDate)#</li>
<li>day: #DatePart("d", todayDate)#</li>
<li>weekday: #DatePart("w", todayDate)#</li>
<li>week: #DatePart("ww", todayDate)#</li>
<li>hour: #DatePart("h", todayDate)#</li>
<li>minute: #DatePart("n", todayDate)#</li>
<li>second: #DatePart("s", todayDate)#</li>
</ul>
</cfoutput>
Day

Description

Determines the day of the month, in a date.

Returns

The ordinal for the day of the month, ranging from 1 to 31.

Category

Date and time functions

Function syntax

```
Day("date")
```

See also

DayOfWeek, DayOfWeekAsString, DayOfYear, DaysInMonth, DaysInYear, FirstDayOfMonth

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

⚠️ Note

You can pass the CreateDate function or Now function as the date parameter of this function; for example: #Day(CreateDate(2001, 3, 3))#

Example
<h3>Day Example</h3>
<cfif IsDefined("FORM.year")>
<p>More information about your date:</p>
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
<cfoutput>
<p>Date: #DateFormat(yourDate)#. <br>
It is #DayOfWeekAsString(DayOfWeek(yourDate))#, <br>
day #DayOfWeek(yourDate)# in the week. <br>
This is day #Day(YourDate)#<br> in the month of #MonthAsString(Month(yourDate))#, <br>
which has #DaysInMonth(yourDate)# days. <br>
We are in week #Week(yourDate)# of #Year(YourDate)#<br>
(day #DayofYear(yourDate)# of #DaysinYear(yourDate)#). <br>
<cfif IsLeapYear(Year(yourDate))>This is a leap year <br>
<cfelse>This is not a leap year</cfif>
</cfoutput>
</cfif>
DayOfWeek

Description

Determines the day of the week, in a date.

Returns

The ordinal for the day of the week, as an integer in the range 1 (Sunday) to 7 (Saturday).

Category

Date and time functions

Function syntax

```
DayOfWeek("date")
```

See also

Day, DayOfWeekAsString, DayOfYear, DaysInMonth, DaysInYear, FirstDayOfMonth

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

⚠️ Note

You can pass the CreateDate function or Now function as the date parameter of this function; for example, #DayOfWeek(Createdate(2001, 3, 3))#

Example
<h3>DayOfWeek Example</h3>
<cfif IsDefined("FORM.year")>
More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
<cfoutput>
<p>Your date, #DateFormat(yourDate)#.
<br>It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day #DayOfWeek(yourDate)# in the week.
<br>This is day #Day(YourDate)# in the month of #MonthAsString(Month(yourDate))#, which has #DaysInMonth(yourDate)# days.
<br>We are in week #Week(yourDate)# of #Year(YourDate)# (day #DayofYear(yourDate)# of #DaysinYear(yourDate)#).
<br><cfif IsLeapYear(Year(yourDate))>This is a leap year
<cfelse>This is not a leap year</cfif>
</cfoutput>
</cfif>
DayOfWeekAsString

Description

Determines the day of the week, in a date, as a string function.

Returns

The day of the week, as a string in the current locale, that corresponds to day_of_week.

Category

Date and time functions, String functions

Function syntax

```
DayOfWeekAsString(day_of_week [, locale])
```

See also

Day, DayOfWeek, DayOfYear, DaysInMonth, DaysInYear, FirstDayOfMonth

History

ColdFusion 8: Added the locale parameter.

ColdFusion MX 7: Changed behavior. The returned string is now in the language of the current locale.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>day_of_week</td>
<td>Integer, in the range 1 (Sunday) - 7 (Saturday).</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>

Example

The following example shows the use of the DayOfWeekAsString function. It is the action page for a form that submits year, month, and day fields.
<h3>DayOfWeekAsString Example</h3>
<cfif IsDefined("FORM.year")>
More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>

<p>Your date, #DateFormat(yourDate)#.
It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day #DayOfWeek(yourDate)# in the week.
This is day #Day(YourDate)# in the month of #MonthAsString(Month(yourDate))#, which has #DaysInMonth(yourDate)# days.
We are in week #Week(yourDate)# of #Year(YourDate)# (day #DayofYear(yourDate)# of #DaysinYear(yourDate)#).
<cfif IsLeapYear(Year(yourDate))>This is a leap year
<cfelse>This is not a leap year</cfif>
</cfif>
DayOfYear

Description

Determines the day of the year, in a date.

Returns

The ordinal value of day of the year, as an integer.

Category

Date and time functions

Function syntax

```
DayOfYear("date")
```

See also

Day, DayOfWeek, DayOfWeekAsString, DaysInMonth, DaysInYear, FirstDayOfMonth

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

This function accounts for leap years. When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

⚠️ Note

You can pass the CreateDate function or Now function as the date parameter of this function; for example, #DayOfYear(CreateDate(2001, 3, 3))#

Example
<h3>DayOfWeek Example</h3>

```coldfusion
<cfif IsDefined("FORM.year")>

More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
<cfoutput>
<p>Your date, #DateFormat(yourDate)#.</p>
<br>It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day #DayOfWeek(yourDate)# in the week.
<br>This is day #Day(yourDate)# in the month of #MonthAsString(Month(yourDate))#, which has #DaysInMonth(yourDate)# days.
<br>We are in week #Week(yourDate)# of #Year(yourDate)# (day #DayofYear(yourDate)# of #DaysInYear(yourDate)#).
<br><cfif IsLeapYear(Year(yourDate))>This is a leap year</cfif>
<cfelse>This is not a leap year</cfif>
</cfoutput>
</cfif>
```
DaysInMonth

Description

Determines the number of days in a month.

Returns

The number of days in the month in Date.

Category

Date and time functions

Function syntax

```plaintext
DaysInMonth("date")
```

See also

Day, DayOfWeek, DayOfWeekAsString, DayOfYear, DaysInYear, FirstDayOfMonth

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

⚠️ Note

You can pass the Now function or the CreateDate function as the date parameter of this function; for example: #DaysInMonth(CreateDate(2001, 3, 3))#

Example
<h3>DaysInMonth Example</h3>
<cfif IsDefined("FORM.year")>
More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
<cfoutput>
<p>Your date, #DateFormat(yourDate)#.</p>
<br>It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day #DayOfWeek(yourDate)# in the week.
<br>This is day #Day(YourDate)# in the month of #MonthAsString(Month(yourDate))#, which has #DaysInMonth(yourDate)# days.
<br>We are in week #Week(yourDate)# of #Year(YourDate)# (day #DayOfYear(yourDate)# of #DaysinYear(yourDate)#).
<br><cfif IsLeapYear(Year(yourDate))>This is a leap year</cfif>
<cfelse>This is not a leap year</cfif>
</cfoutput>
</cfif>
**DaysInYear**

**Description**

Determines the number of days in a year.

**Returns**

The number of days in a year.

**Category**

Date and time functions

**Function syntax**

```
DaysInYear("date")
```

**See also**

Day, DayOfWeek, DayOfWeekAsString, DayOfYear, DaysInMonth, FirstDayOfMonth, IsLeapYear

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

**Usage**

DaysInYear accounts for leap years. When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a numeric representation of a date/time object.

⚠️ **Note**

You can pass theCreateDate function or the Now function as the date parameter of this function; for example: 

```
#DaysInYear(CreateDate(2001, 3, 3))#
```

**Example**
<h3>DaysInYear Example</h3>
<cfif IsDefined("FORM.year")>
More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
<cfoutput>
<p>Your date, #DateFormat(yourDate)#.
<br>It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day
    #DayOfWeek(yourDate)# in the week.
<br>This is day #Day(YourDate)# in the month of
    #MonthAsString(Month(yourDate))#, which has
    #DaysInMonth(yourDate)# days.
<br>We are in week #Week(yourDate)# of #Year(yourDate)# (day
    #DayofYear(yourDate)# of #DaysinYear(yourDate)#).
</cfif>
**DE**

**Description**

Escapes any double-quoted marks in the parameter and wraps the result in double-quoted marks.

**Returns**

Parameter, surrounded by double-quoted marks, with any inner double-quoted marks escaped.

**Category**

Dynamic evaluation functions

**Function syntax**

```
DE(string)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>String to evaluate, after delay</td>
</tr>
</tbody>
</table>

**Usage**

The `DE` function postpones evaluation of a string that is passed as a parameter to the `IIF`, `Evaluate`, or `PrecisionEvaluate` functions. This function is especially useful with the `IIF` function, which automatically evaluates its second and third parameters as expressions. You can use the `DE` function to prevent the function from evaluating a string parameter that is to be output as a variable, and must not be treated as an expression. The following example shows this use; it uses `IIF` to alternate table-row background colors, white and gray, and uses the `DE` function to prevent ColdFusion from evaluating the color strings.

```cfoutput
<table border="1" cellpadding="3">
<cfloop index="i" from="1" to="10">
  <tr bgcolor="#IIF( i mod 2 eq 0, DE("white"), DE("gray") )#">
    <td>
      hello #i#
    </td>
  </tr>
</cfloop>
</table>
</cfoutput>
```

The `DE` function does not delay evaluation of variable names that are surrounded by number signs (#). The
ColdFusion function evaluates the variable regardless of whether the `DE` function is present. The following example shows how you can use the `DE` function and number signs together, and shows how the function works with an `IIF` function:

```coldfusion
<cfoutput>
<cfset var1="Blue">
<cfset var2="Green">
<cfset myresult=IIf( 1 eq 2, DE(#Var1#), DE(#Var2#))>
The expression is #myresult#
</cfoutput>
```

ColdFusion processes this code as follows:

1. ColdFusion sets the variables `var1` and `var2` to be the strings Blue and Green.
2. In the fourth line, ColdFusion evaluates the variables surrounded by number signs first, replacing them with the strings Blue and Green, the values of the variables.
3. The `IIF` function evaluates the test expression, determines that it is False, and then evaluates the third parameter.
4. The third parameter is a `DE` function, which takes the string Green and surrounds it in quotation marks.
5. The `IIF` function returns the string "Green", including the quotation marks.
6. The `cfset` tag gets the expression result="Green", and sets the value of the `myresult` variable to the string Green.
7. ColdFusion evaluates `#myresult#` in the output text, replaces the variable with its value, the string Green, and displays the result.
<!--- This example shows the use of DE and Evaluate --->
<h3>DE Example</h3>
<cfif IsDefined("FORM.myExpression")>
  <cftry>
    <!--- Show the expression and the results of evaluating it --->
    <cfoutput>
      <h3>Evaluate the Expression #FORM.MyExpression#</h3>
      The code:<br>
      #Evaluate(FORM.myExpression)#
      <br><br>
      The result:<br>
      <cfoutput>
        #Evaluate(FORM.myExpression)#
      </cfoutput>
    </cfoutput>
  </cftry>
  <h3>Use DE to prevent the Evaluate function from evaluating</h3>
  The code:<br>
  #Evaluate(DE(FORM.MyExpression))#<br><br>
  The result:<br>
  <cfoutput>
    #Evaluate(DE(FORM.MyExpression))#
  </cfoutput>
  <!--- Error handling code for bad expressions and any other error. --->
  <cfcatch type = "Any">
    <!--- the message to display --->
    <h3>Sorry, there's been an <B>Error</B>. Try a simple expression, such as "2+2".</h3>
    <cfoutput>
      <p>#cfcatch.message#</p>
    </cfoutput>
  </cfcatch>
</cftry>
<cfif>
  <h3>Enter any valid ColdFusion expression</h3>
  <cfform>
    <cfinput name="myExpression" type="Text" size="40">
    <cfinput type="submit" name="submitit">
  </cfform>
DecimalFormat

Description

Converts a number to a decimal-formatted string.

Returns

A *number* as a string formatted with two decimal places and a thousands separator.

Category

*Display and formatting functions*

Function syntax

```plaintext
DecimalFormat(number)
```

See also

*DollarFormat, NumberFormat*

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Number to format</td>
</tr>
</tbody>
</table>

Example

```plaintext
<h3>DecimalFormat Function</h3>
<p>Returns a number to two decimal places.</p>
<p>
<cfloop FROM = 1 TO = 20 INDEX = "counter">
  <cfoutput>
    #counter# * Square Root of 2:
    #DecimalFormat(counter * sqr(2))#
  </cfoutput>
  <br>
</cfloop>
```
DecodeForHTML

Description

Decodes an HTML encoded string.

Returns

Decoded HTML string.

Category

Display and formatting functions.

Function syntax

```
DecodeForHTML(String encodedinput)
```

See also

Canonicalize, EncodeForHTMLAttribute, EncodeForJavaScript, EncodeForCSS, EncodeForURL

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The encoded string to decode.</td>
</tr>
</tbody>
</table>

Example

```
<cif isDefined("form.submit")>
 <b>
  Output:<cfoutput >#DecodeForHTML(form.encodedUserName)#</cfoutput>
 </b>
<celse>
  <cfset form.username="/">
 </cif>
</cfform>
```

```
 <cinput name="encodedUserName" type="text" value="#form.encodedUserName#">
 <cinput name="submit" type="submit" value="Submit">
</cform>
```
**DecodeFromURL**

**Description**

Decodes an encoded HTML URL string.

**Returns**

Decoded HTML URL string.

**Category**

Display and formatting functions.

**Function syntax**

```
DecodeFromURL(String encodedinput)
```

**See also**

Canonicalize, EncodeForHTMLAttribute, EncodeForJavaScript, EncodeForCSS, EncodeForURL

**History**

ColdFusion 10: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The encoded URL string to decode.</td>
</tr>
</tbody>
</table>

**Example**

```
<cfset string = "http://www.adobe.com/" >
<cfset urlencoded = encodeforURL(string)>  
<cfset urldecoded = decodefromUrl(urlEncoded)>  
<cfoutput>
  String: #string# <br/>
  URL Encoded: #urlencoded#<br/>  
  URL Decoded: #urldecoded#<br/>
</cfoutput>
```
**DecrementValue**

**Description**

Decrement the integer part of a number.

**Returns**

Integer part of *number*, decremented by one.

**Category**

*Mathematical functions*

**Function syntax**

```
DecrementValue(number)
```

**See also**

*IncrementValue*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Number to decrement</td>
</tr>
</tbody>
</table>

**Example**

```
<h3>DecrementValue Example</h3>
<p>Returns the integer part of a number decremented by one.</p>
<p>DecrementValue(0):
<cfoutput>#DecrementValue(0)#</cfoutput></p>
<p>DecrementValue("1"):
<cfoutput>#DecrementValue("1")#</cfoutput></p>
<p>DecrementValue(123.35):
<cfoutput>#DecrementValue(123.35)#</cfoutput></p>
```
Decrypt

Description

Decrypts a string that is encrypted using a standard encryption technique, including strings encrypted by the Encrypt function.

Returns

An unencrypted string.

Category

Security functions, String functions

Function syntax

```
Decrypt(encrypted_string, key[, algorithm, encoding, IVorSalt, iterations])
```

See also

Duplicate, Encrypt

History

ColdFusion 8: Added support for encryption using the RSA BSafe Crypto-J library on Enterprise Edition.

ColdFusion MX 7.01: Added the IVorSalt and iterations parameters.

ColdFusion MX 7: Added the algorithm and encoding parameters.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>encrypted_string</td>
<td>String to decrypt.</td>
</tr>
<tr>
<td>key</td>
<td>String. For the CFMX_COMPAT algorithm, the seed that was used to encrypt the string; for all other algorithms, the string generated by the generateSecretKey() method.</td>
</tr>
</tbody>
</table>
| **algorithm** | (Optional) The Enterprise Edition of ColdFusion installs the RSA BSafe Crypto-J library, which provides FIPS-140 Compliant Strong Cryptography. For a list of algorithms, see the Encrypt function. The Standard Edition of ColdFusion installs a cryptography library with the following algorithms:

- CFMX_COMPAT: the algorithm used in ColdFusion MX and prior releases. This algorithm is the least secure option (default).
- AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197.
- BLOWFISH: the Blowfish algorithm defined by Bruce Schneier.
- DESEDE: the "Triple DES" algorithm defined by NIST FIPS-46-3.

If you install a security provider with additional cryptography algorithms, you can also specify any of its string encryption and decryption algorithms. |
| **encoding** | (Optional; if you specify this parameter, also specify the algorithm parameter.) The binary encoding used to represent the data as a string. Must be the same as the algorithm used to encrypt the string.

- Base64: the Base64 algorithm, as specified by IETF RFC 2045.
- Hex: the characters A-F and 0-9 represent the hexadecimal byte values.
- UU: the UNIX standard UUEncode algorithm (default). |
| **IVorSalt** | (Optional) Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, also specify the algorithm parameter.

- For Block Encryption Algorithms: The binary Initialization Vector value to use with the algorithm. The algorithm must contain a Feedback Mode other than ECB. This must be a binary value that is exactly the same size as the algorithm block size.
- For Password Based Encryption Algorithms: The binary Salt value to transform the password into a key. |
| iterations | (Optional) The number of iterations to transform the password into a binary key. Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, also specify the algorithm parameter with a Password Based Encryption (PBE) algorithm. Do not specify this parameter for Block Encryption Algorithms. Use the same value to encrypt and decrypt the data. |

**Usage**

This function uses a symmetric key-based algorithm, in which the same key is used to encrypt and decrypt a string. The parameter values must match the values used to encode string. The security of the encrypted string depends on maintaining the secrecy of the key. ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, Adobe cannot provide technical support for third-party security providers.

**Example**

```csharp
<h3>Decrypt Example</h3>

<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>
  <cfscript>
    /* GenerateSecretKey does not generate key for the CFMX_COMPAT algorithm, so use the key from the form. */
    if (Form.myAlgorithm EQ "CFMX_COMPAT")
      theKey=Form.MyKey;
    // For all other encryption techniques, generate a secret key.
    else
      theKey=generateSecretKey(Form.myAlgorithm);
    //Encrypt the string
    encrypted=encrypt(Form.myString, theKey, Form.myAlgorithm, Form.myEncoding);
    //Decrypt it
    decrypted=decrypt(encrypted, theKey, Form.myAlgorithm, Form.myEncoding);
  </cfscript>

  <!--- Display the values used for encryption and decryption, and the results. --->
  <cfoutput>
    <b>The algorithm:</b> #Form.myAlgorithm#
    <br>
    <b>The key:</b> #theKey#
    <br>
    <b>The string:</b> #Form.myString# <br>
    <br>
    <b>Encrypted:</b> #encrypted# <br>
    <br>
    <b>Decrypted:</b> #decrypted# <br>
  </cfoutput>
</cfif>

<!--- The input form.--->
```
<form action="#CGI.SCRIPT_NAME#" method="post">
    <b>Select the encoding</b><br>
    <select size="1" name="myEncoding">
        <option selected>UU</option>
        <option>Base64</option>
        <option>Hex</option>
    </select><br>

    <br>
    <b>Select the algorithm</b><br>
    <select size="1" name="myAlgorithm">
        <option selected>CFMX_COMPAT</option>
        <option>AES</option>
        <option>DES</option>
        <option>DESEDE</option>
    </select><br>

    <br>
    <b>Input your key</b> (used for CFMX_COMPAT encryption only)<br>
    <input type = "Text" name = "myKey" value = "MyKey"><br>

    <br>
    <b>Enter string to encrypt</b><br>
    <textarea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">
        This string is encrypted (you can replace it with more typing).
    </textarea><br>
</form>
<input type = "Submit" value = "Encrypt my String">
</form>
DecryptBinary

Description

Decrypts encrypted binary data with the specified key, value, algorithm, salt, and iterations.

Returns

Unencrypted binary data.

Category

Security functions, String functions

Function syntax

DecryptBinary(bytes, key[, algorithm, IVorSalt, iterations])

See also

Duplicate, Encrypt, Decrypt

History

ColdFusion 8: Added support for encryption using the RSA BSafe Crypto-J library on Enterprise Edition.

ColdFusion MX 7.01: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bytes</td>
<td>Binary data to decrypt.</td>
</tr>
<tr>
<td>key</td>
<td>String. For the CFMX_COMPAT algorithm, the seed that was used to encrypt the binary data; for all other algorithms, the string generated by the generateSecretKey() method.</td>
</tr>
<tr>
<td>parameter</td>
<td>description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>algorithm</td>
<td>(Optional) The Enterprise Edition of ColdFusion installs the RSA BSafe Crypto-J library, which provides FIPS-140 Compliant Strong Cryptography. For a list of algorithms, see the Encrypt function. The Standard Edition of ColdFusion installs a cryptography library with the following algorithms:</td>
</tr>
<tr>
<td></td>
<td>• CFMX_COMPAT: the algorithm used in ColdFusion MX and prior releases. This algorithm is the least secure option (default).</td>
</tr>
<tr>
<td></td>
<td>• AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197.</td>
</tr>
<tr>
<td></td>
<td>• BLOWFISH: the Blowfish algorithm defined by Bruce Schneier.</td>
</tr>
<tr>
<td></td>
<td>• DES: the Data Encryption Standard algorithm defined by NIST FIPS-46-3.</td>
</tr>
<tr>
<td></td>
<td>• DESEDE: the &quot;Triple DES&quot; algorithm defined by NIST FIPS-46-3.</td>
</tr>
<tr>
<td></td>
<td>If you install a security provider with additional cryptography algorithms, you can also specify any of its string encryption and decryption algorithms.</td>
</tr>
<tr>
<td>IVorSalt</td>
<td>(Optional) Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, also specify the algorithm parameter.</td>
</tr>
<tr>
<td></td>
<td>• For Block Encryption Algorithms: The binary Initialization Vector value to use with the algorithm. The algorithm must contain a Feedback Mode other than ECB. This must be a binary value that is exactly the same size as the algorithm block size.</td>
</tr>
<tr>
<td></td>
<td>• For Password Based Encryption Algorithms:- This is the binary Salt value to transform the password into a key.</td>
</tr>
<tr>
<td>iterations</td>
<td>(Optional) The number of iterations to transform the password into a binary key. Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, also specify the algorithm parameter with a Password Based Encryption (PBE) algorithm. Do not specify this parameter for Block Encryption Algorithms. Use the same value to encrypt and decrypt the data.</td>
</tr>
</tbody>
</table>

**Usage**

This function uses a symmetric key-based algorithm, in which the same key is used to encrypt and decrypt data. The parameter values must match the values used to encode the string. The security of the encrypted string depends on maintaining the secrecy of the key. ColdFusion uses the Java Cryptography Extension (JCE) and
installs a Sun Java runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, Adobe cannot provide technical support for third-party security providers.

Example
<h3>DecryptBinary Example</h3>

```coldfusion
<cfif IsDefined("Form.myfile")>
  <cffile file="#Form.myfile#" action="readBinary" variable="myData">
  <cfscript>
    /* GenerateSecretKey does not generate key for the CFMX_COMPAT algorithm, so use the key from the form. */
    if (Form.myAlgorithm EQ "CFMX_COMPAT")
      theKey=Form.MyKey;
    // For all other encryption techniques, generate a secret key.
    else
      theKey=generateSecretKey(Form.myAlgorithm);
    //Encrypt the string
    encrypted=encryptBinary(myData, theKey, Form.myAlgorithm);
    //Decrypt it
    decrypted=decryptBinary(encrypted, theKey, Form.myAlgorithm);
  </cfscript>
  <cfset encfile="#Form.myfile# & "_enc">
  <cfset decfile="#Form.myfile# & "_dec">
  <cffile file="#encfile#" action="write" output="#encrypted#">
  <cffile file="#decfile#" action="write" output="#decrypted#">
  <!--- Display the values used for encryption and decryption, and the results. --->
  <cfoutput>
    <b>The algorithm:</b> #Form.myAlgorithm#<br/>
    <b>The key:</b> #theKey#<br/>
    <br/>
    <b>The InputFile:</b> #Form.myfile#<br/>
    <br/>
    <b>Encrypted:</b> #encfile#<br/>
    <br/>
    <b>Decrypted:</b> #decfile#<br/>
  </cfoutput>
</cfif>
```

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
  <b>Select the algorithm</b><br/>
  <select size="1" name="myAlgorithm">
    <option selected>CFMX_COMPAT</option>
    <option>AES</option>
    <option>DES</option>
    <option>DESEDE</option>
  </select><br/>
  <br/>
  <b>Input your key</b> (used for CFMX_COMPAT encryption only)<br/>
  <input type = "Text" name = "myKey" value = "MyKey"><br/>
  <br/>
  <b>Enter filename to encrypt</b><br/>
  <input type="text" name="myfile" value="Enter the path of the file to encrypt"><br/>
  <input type = "Submit" value = "Encrypt file ">
</form>
DeleteClientVariable

Description

Deletes a client variable. (To test for the existence of a variable, use IsDefined.)

Returns

True, if the variable is successfully deleted; false, otherwise.

Category

Other functions

Function syntax

```
DeleteClientVariable("name")
```

See also

GetClientVariablesList

History

ColdFusion MX: Changed behavior: if the variable is not present, this function now returns False. (In earlier releases, it threw an error.)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of a client variable to delete, surrounded by double-quotation marks</td>
</tr>
</tbody>
</table>

Example
<!--- This view-only example shows DeleteClientVariable --->
<h3>DeleteClientVariable Example</h3>

<p>This view-only example deletes a client variable called "User_ID", if it exists in the list of client variables returned by GetClientVariablesList.</p>

<p>This example requires the existence of an Application.cfm file and client management to be in effect.</p>

<!----
<cfset client.somevar = ""
<cfset client.user_id = ""
<p>Client variable list:<cfoutput>#GetClientVariablesList()#</cfoutput>
<cfif ListFindNoCase(GetClientVariablesList(), "User_ID") is not 0>
    <cfset temp = DeleteClientVariable("User_ID")>
    <p>Was variable "User_ID" Deleted? <cfoutput>#temp#</cfoutput>
</cfif>
<p>Amended Client variable list:<cfoutput>#GetClientVariablesList()#</cfoutput>
</cfoutput>---->
DeserializeJSON

Description

Converts a JSON (JavaScript Object Notation) string data representation into CFML data, such as a CFML structure or array.

Returns

The data value in ColdFusion format: a structure, array, query, or simple value.

Category

Conversion functions

Syntax

DeserializeJSON(JSONVar[, strictMapping, useCustomSerializer])

See also


History

ColdFusion 11: Added the attribute, useCustomSerializer.

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSONVar</td>
<td>A string that contains a valid JSON construct, or variable that represents one.</td>
</tr>
<tr>
<td>strictMapping</td>
<td>A Boolean value that specifies whether to convert the JSON strictly, as follows:</td>
</tr>
<tr>
<td></td>
<td>• true: (Default) Convert the JSON string to ColdFusion data types that correspond directly to the JSON data types.</td>
</tr>
<tr>
<td></td>
<td>• false: Determine if the JSON string contains representations of ColdFusion queries, and if so, convert them to queries.</td>
</tr>
<tr>
<td>useCustomSerializer</td>
<td>true/false. Whether to use the customSerializer or not. The default value is true. The custom serializer will always be used for deserialization. If false, the JSON deserialization will be done using the default ColdFusion behavior.</td>
</tr>
</tbody>
</table>
Usage

This function is useful any time a ColdFusion page receives data as JSON strings. It is useful in ColdFusion applications that use Ajax to represent data on the client browser, and lets you consume on the server JSON format data from the client-side Ajax JavaScript. You can also use it on pages that get data from services that supply data as JavaScript function calls with JSON parameters; the example shows this use case. The DeserializeJSON function converts each JSON data type directly into the equivalent ColdFusion data type, as follows:

- If the strictMapping parameter is true (the default), all JSON objects become CFML structures.
- If the strictMapping parameter is false, ColdFusion determines if JSON objects represent queries and, if so, converts them to ColdFusion query object. All other JSON objects become ColdFusion structures. The DeserializeJSON function recognizes a JSON structure as a query and converts it properly if the structure uses either of the two query representation formats described in the SerializeJSON reference.
- JSON Arrays, Strings, and Numbers become ColdFusion arrays, strings, and numbers.
- The JSON null value becomes the string null.
- JSON string representations of a dates and times remain strings, but ColdFusion date/time handling code can recognize them as representing dates and times.

Example

This example displays weather information from a JSON-format data feed that is generated by the example for the SerializeJSON function. Similar code might consume data that is exported as a JavaScript page. The feed is in the form of a JavaScript function call where the parameter is a JSON string that contains the feed data. The example does the following operations:

1. Uses a cfhttp tag to get the feed (in the cfhttp.fileContent variable).
2. Strips the function call wrapper from the text.
3. Uses the IsJSON function to check whether the result of the previous step is a valid JSON format string. If it is not, it displays a message and stops processing.
4. If the string is valid JSON text, uses the DeserializeJSON function to convert the string to a ColdFusion variable; in this case, a structure that contains two arrays that represent a ColdFusion query. The first array has the query column names, the second has the query data.
5. Parses the object and displays the contents of its arrays.

To run this example, put this file and the example for the SerializeJSON function in an appropriate location under your ColdFusion web root, replace the URL with the correct URL for the serialization example, and run this page.
<!--- Get the JSON Feed --->
<cfhttp url="http://localhost:8500/My_Stuff/Ajax/Books/CreateJSON_NEW.cfm">

<!--- JSON data is sometimes distributed as a JavaScript function. The following REREplace functions strip the function wrapper. --->
<cfset theData=REREplace(cfhttp.FileContent, "^[[:word:]]*\s*\(.*\$", "">
<cfset theData=REREplace(theData, "\s*\$", "">

<!--- Test to make sure you have JSON data. --->
<cfif !IsJSON(theData)>
<h3>The URL you requested does not provide valid JSON</h3>
<cfdump var="#theData#">
</cfif>

<!--- If the data is in JSON format, deserialize it. --->
<cfelse>
<cfset cfData=DeserializeJSON(theData)>

<!--- Parse the resulting array or structure and display the data. --->
In this case, the data represents a ColdFusion query that has been serialized by the SerializeJSON function into a JSON structure with two arrays: an array column names, and an array of arrays, where the outer array rows correspond to the query rows, and the inner array entries correspond to the column fields in the row. --->

<!--- First, find the positions of the columns in the data array. --->
<cfset colList=ArrayToList(cfData.COLUMNS)>
<cfset cityIdx=ListFind(colList, "City")>
<cfset tempIdx=ListFind(colList, "Temp")>
<cfset fcstIdx=ListFind(colList, "Forecasts")>

<!--- Now iterate through the DATA array and display the data. --->
<cfoutput>
<cfloop index="i" from="1" to="#ArrayLen(cfData.DATA)#">
Current Temperature: #cfData.DATA[i][tempIdx]#<br>
<br>Forecasts<br>
<cfloop index="j" from="1" to="#ArrayLen(cfData.DATA[i][fcstIdx])#">
Day #j#<br>
Outlook: #cfData.DATA[i][fcstIdx][j].WEATHER#<br>
High: #cfData.DATA[i][fcstIdx][j].HIGH#<br>
Low: #cfData.DATA[i][fcstIdx][j].LOW#<br>
</cfloop>
</cfloop>
</cfoutput>
</cfif>
DirectoryCopy

Description

Copies the contents of a directory to a destination directory.

Returns

Nothing

History

ColdFusion 10: Added this function.

Syntax

directoryCopy (source, destination[, recurse][, filter])

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Absolute pathname of directory from which you copy content.</td>
</tr>
<tr>
<td>destination</td>
<td>Path of the destination directory. If not an absolute path, it is relative to the source directory.</td>
</tr>
<tr>
<td>recurse</td>
<td>By default, false. If true, copies the subdirectories.</td>
</tr>
<tr>
<td>filter</td>
<td>File extension filter applied, for example, *.cfm.</td>
</tr>
</tbody>
</table>

Example

directoryCopy(sourceDirExists, destDirExists, true,"*.cfm")
**DirectoryCreate**

**Description**

Creates on-disk or in-memory directory.

**Category**

*System functions*

**Function Syntax**

`DirectoryCreate(path)`

**See Also**

`DirectoryDelete`, `DirectoryExists`, `DirectoryList`, `DirectoryRename`

**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Absolute path of the directory to be created. Alternatively, you can specify IP address, as in the following example: <code>DirectoryCreate(&quot;//12.3.12.3.123/c_drive/test&quot;)</code></td>
</tr>
</tbody>
</table>

**Usage**

Ensure that you have the required permissions to run this function.

**Example**

The following code illustrates how to create a directory:
<h2>DirectoryCreate Example</h2>
<h3>Enter a directory to create.</h3>

```cfml
<cfform action = "directorycreate.cfm" method="post" preservedata="true" >
  <cfinput type = "text" required="true" name = "createDirectory">
  <br>
  <cfinput type = "submit" value="submit" name = "submit">
</cfform>

<cfif IsDefined("FORM.createDirectory")>
  <cfif FORM.createDirectory is not ">
    <cfset createDirectory = FORM.createDirectory>
    <cftry>
      <cfset DirectoryCreate(createDirectory)>
      <cfoutput><b>Directory #createDirectory# successfully created.</b></cfoutput>
    </cfcatch>
    <b>Error Message:</b><cfoutput>#cfcatch.message#</cfoutput><br/>
    <b>Error Detail:</b><cfoutput>#cfcatch.Detail#</cfoutput>
  </cfcatch>
</cfif>
</cfif>
DirectoryDelete

Description

Deletes on-disk or in-memory directory.

Category

System functions

Function syntax

DirectoryDelete(path [, recurse])

See also

DirectoryCreate, DirectoryExists, DirectoryList, DirectoryRename

History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Absolute path of the directory to be deleted. Alternatively, you can specify IP address, as in the following example: {{DirectoryDelete(&quot;//12.3.123.123/c_drive/test&quot;); }}.</td>
</tr>
<tr>
<td>recurse</td>
<td>This is an optional parameter and the default value is false. If true, the directory and the sub-directories are deleted. If the directory (being deleted) has sub-directories and you set recurse to false, an exception occurs.</td>
</tr>
</tbody>
</table>

Usage

Ensure that you have the required permissions to run this function.

Example
<h2>DirectoryDelete Example</h2>

<form action = "directoryDelete.cfm" method="post">
  <label for="delDirectory">Directory Path: </label><input type = "text" id="delDirectory" name = "delDirectory">
  <br>
  <label for="recurse">Recurse: </label><input id="recurse" type="checkbox" value="recurse" name="recurse">
  <br>
  <input type = "submit" value="submit" name = "submit" onclick='return confirm("Are you sure !!!");'>
</form>

<cfif IsDefined("FORM.delDirectory")>
  <cfif FORM.delDirectory is not ">
    <cfset delDirectory = FORM.delDirectory>
    <cfset recurse = false>
    <cfif isDefined("FORM.recurse")>
    <cfset recurse = true>
  </cfif>
  <cftry>
    <cfset DirectoryDelete(delDirectory, recurse)>
    <cfoutput><p>Directory <b>#delDirectory#</b> has been deleted.</cfoutput>
    <cfcatch>
      <b>Error Message:</b><cfoutput>#cfcatch.message#</cfoutput><br/>
      <b>Error Detail:</b><cfoutput>#cfcatch.Detail#</cfoutput>
    </cfcatch>
  </cftry>
</cfif>
</cfif>
**DirectoryExists**

**Description**

Determines whether on-disk or in-memory directory exists.

**Returns**

Yes, if the specified directory exists; No, otherwise.

**Category**

*System functions*

**Function syntax**

```
DirectoryExists(absolute_path)
```

**See also**

DirectoryCreate, DirectoryDelete, DirectoryList, DirectoryRename

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolute_path</td>
<td>An absolute on-disk or in-memory path. Alternatively, you can specify IP address as in the following example: DirectoryExists(&quot;/12.3.123.123/c_drive/test&quot;);</td>
</tr>
</tbody>
</table>

**Example**
<h3>DirectoryExists Example</h3>
Enter a directory to check for existence.

<form action="directoryexists.cfm" method="post">
  <input type="text" name="yourDirectory">
  <br>
  <input type="submit" name="submit">
</form>

<cfif IsDefined("FORM.yourDirectory")>
  <cfif FORM.yourDirectory is not ">
    <cfset yourDirectory = FORM.yourDirectory>
    <cfoutput>
      <p>Your directory exists. Directory name: #yourDirectory#</p>
    </cfoutput>
  </cfifelse>
  <p>Your directory does not exist.</p>
</cfif>
</cfif>
**DirectoryList**

**Description**
Lists the contents of on-disk or in-memory directory. Also lists the contents of the sub-directories if `recurse` is set to `true`.

**Returns**
Contents of the directory based on the parameter `listInfo`:

- If `listInfo=\"query\"`, query object
- If `listInfo=\"name\"`, array of names
- If `listInfo=\"path\"`, array of path

**Category**
*System functions*

**Function Syntax**
```
```

**See Also**
`DirectoryCreate`, `DirectoryDelete`, `DirectoryExists`, `DirectoryRename`

**History**
ColdFusion 11: Added the `type` argument.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>path</code></td>
<td>The absolute path of the directory for which to list the contents. Alternatively, you can specify IP address as in the following example: <code>DirectoryList(&quot;//12.3.123.123/c_drive/test&quot;)</code>.</td>
</tr>
<tr>
<td><code>recurse</code></td>
<td>Whether ColdFusion performs the action on subdirectories: If <code>true</code>, contents of all subdirectories are also listed.</td>
</tr>
</tbody>
</table>
| `listInfo` | • `name`: returns an array of names of files and directories.  
|            | • `path`: returns an array of paths of files and directories.  
|            | • `query`: returns a query. |
File extension filter applied to returned names, for example, *.cfm. Multiple filters can be applied by using a pipe delimiter. For example: *.beer|*.cookies

Also, you can pass a function in the filter argument:

```cfscript
boolean function filterBySize(path, type, extension) {
    var sizeLimit = 1024 * 100;
    // more than 10 KB
    var extensionList = "jpg,jpeg,gif,png";
    if(type is "dir") return false;

    if(listFindNoCase(extensionList, extension)) {
        var fileInfo = getInfo(path);
        var size = fileInfo.size;
        if(size >= sizeLimit) return true;
    }
    return false;
}

filteredResults = directorylist(expandPath("."),
true, "path", filterBySize);
writeDump(filteredResults);
</cfscript>

The arguments of the passed functions must have:

- path: The file path
- type: The values (file or dir)
- extension: The file extension if any otherwise an empty string

Additionally, it can also accept the instances of Java `FileFilter` Objects.
### sort

Query columns by which to sort a directory listing. Delimited list of columns from query output. To qualify a column, use one of the following values:

- `asc`: ascending (a to z) sort order.
- `dec`: descending (z to a) sort order.

For example: `sort = "directory ASC, size DESC, datelastmodified"

<table>
<thead>
<tr>
<th>type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>file</code></td>
<td>includes only filenames.</td>
</tr>
<tr>
<td><code>dir</code></td>
<td>includes only directory names.</td>
</tr>
<tr>
<td><code>all</code></td>
<td>includes both filenames and directory names.</td>
</tr>
</tbody>
</table>

### Usage

Ensure that you have the required permissions to run this function.

### Example

The following code dumps the contents of a directory:
<h2>DirectoryList Example</h2>
<h3>Enter a directory for Listing.</h3>

```coldfusion
<cfif IsDefined("FORM.listDirectory")>
  <cfif FORM.listDirectory is not ">
    <cfset listDirectory = FORM.listDirectory>
    <cfset recurse = false>
    <cfset listInfo = FORM.listInfo>
    <cfset filter = FORM.filter>
      
      <cftry>
        <cfset res= DirectoryList(listDirectory, recurse, listInfo, filter)>
        <cfoutput><b>Content of Directory #listDirectory#: </b></cfoutput>
        <cfdump var="#res#">
        <cfcatch>
          <b>Error Message: </b><cfoutput>#cfcatch.message#</cfoutput><br/>
          <b>Error Detail: </b><cfoutput>#cfcatch.Detail#</cfoutput>
        </cfcatch>
      </cftry>
    </cfif>
  </cfif>
</cfif>
```

```html
<cfif FORM.listDirectory is not ">
  <cfset listDirectory = FORM.listDirectory>
  <cfset recurse = false>
  <cfset listInfo = FORM.listInfo>
  <cfset filter = FORM.filter>

  <label for="listDirectory">Directory Path: </label><cfinput type = "text" id="listDirectory" name = "listDirectory">
  <br />
  <label for="recurse">Recurse: </label><cfinput id="recurse" type="checkbox" value="recurse" name="recurse">
  <br />
  <label for="listInfo">List Info: </label>
  <cfselect name="listInfo" id="listInfo">
    <option value="name">name</option>
    <option value="path">path</option>
    <option value="query">query</option>
  </cfselect>
  <br />
  <label for="filter">Filter: </label><cfinput id="filter" type="text" value="" name="filter">
  <br />
  <input type = "submit" value="submit" name = "submit">
</cfif>
```
**DirectoryRename**

**Description**

Renames on-disk or in-memory directory.

**Category**

*System functions*

**Function Syntax**

`DirectoryRename(currentName, newName)`

**See Also**

`DirectoryCreate`, `DirectoryDelete`, `DirectoryExists`, `DirectoryList`

**Usage**

Ensure that you have the required permissions to run this function.

**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>currentName</code></td>
<td>Absolute path of the directory to be renamed. Alternatively, you can specify IP address, for example, <code>DirectoryRename(&quot;//12.3.123.123/c_drive/test&quot;);</code></td>
</tr>
<tr>
<td><code>newName</code></td>
<td>Name with which the directory has to be renamed.</td>
</tr>
</tbody>
</table>

**Example**
<h2>DirectoryRename Example</h2>
<h3>Enter a directory to rename.</h3>
<cfform action="directoryRename.cfm" method="post" preservedata="true">
  <label for="renameDirectory">Directory to rename: </label>
  <cfinput required="true" type="text" id="renameDirectory" name="renameDirectory">
  <br/>
  <label for="newName">New Name: </label>
  <cfinput required="true" type="text" id="newName" name="newName">
  <br/>
  <input type="submit" value="submit" name="submit">
</cfform>

<cfif IsDefined("FORM.renameDirectory") and IsDefined("FORM.newName")>
  <cfif FORM.renameDirectory is not "" and FORM.newName is not "">
    <cfset renameDirectory = FORM.renameDirectory>
    <cfset newName = FORM.newName>
    <cftry>
      <cfset DirectoryRename(renameDirectory, newName)>
      <cfoutput><b>Directory #renameDirectory# renamed to newName</b></cfoutput>
      <cfcatch>
        <b>Error Message:</b><cfoutput>#cfcatch.message#</cfoutput><br/>
        <b>Error Detail:</b><cfoutput>#cfcatch.Detail#</cfoutput><br/>
      </cfcatch>
    </cftry>
  </cfif>
</cfif>
**DollarFormat**

**Description**

Formats a string in U.S. format. (For other currencies, use **LSCurrencyFormat** or **LSEuroCurrencyFormat**.)

**Returns**

A number as a string, formatted with two decimal places, thousands separator, and dollar sign. If *number* is negative, the return value is enclosed in parentheses. If *number* is an empty string, returns zero.

**Category**

*Display and formatting functions*

**Function syntax**

```
DollarFormat(number)
```

**See also**

**DecimalFormat**, **NumberFormat**

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Number to format</td>
</tr>
</tbody>
</table>

**Example**

```html
<!--- This example shows the use of DollarFormat --->
...
<h3>DollarFormat Example</h3>
<cfloop from = 8 to = 50 index = counter>
  <cfset full = counter>
  <cfset quarter = counter + (1/4)>
  <cfset half = counter + (1/2)>
  <cfset threefourth = counter + (3/4)>
  <cfoutput>
    bill #DollarFormat(full)# #DollarFormat(quarter)# #DollarFormat(half)# #DollarFormat(threefourth)#
    18% tip #DollarFormat(full * (18/100))# #DollarFormat(quarter * (18/100))# #DollarFormat(half * (18/100))# #DollarFormat(threefourth * (18/100))#
  </cfoutput>
</cfoutput>
</cfloop>
...
DotNetToCFType

Description

Explicitly converts a value returned by a .NET method to the corresponding ColdFusion data type.

Returns

A ColdFusion data value.

Category

Structure functions, System functions

Function syntax

DotNetToCFType(variable_name)

See also

Converting between .NET and ColdFusion data types in Using .NET classes in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable_name</td>
<td>Name of the .NET variable to convert</td>
</tr>
</tbody>
</table>

Usage

For detailed information on when and why you use this function, see Working with complex .NET data types in Using .NET classes in the Developing ColdFusion Applications.

Example

The following example creates a .NET System.Data.DataTable object and converts it to a ColdFusion query.
<cfobject action="create" name="sqlCommandObject"
    class="System.Data.SqlClient.SqlCommand" type=".Net"
    assembly="#assemblyList#">
    <cfset sqlCommandObject.init("SELECT [ID], [FriendlyName] FROM [Batch]",
        sqlConnectionObject)>
    <cfset sqlDataReaderObject = sqlCommandObject.ExecuteReader()>
    <cfset dataTable = createObject(".net", "System.Data.DataTable",
        assemblyList)>
    <!-- populate the datatable -->
    <cfset dataTable.load(sqlDataReaderObject)>
    <!-- convert to cfquery -->
    <cfset myquery=DotNetToCFType(dataTable)>
**Duplicate**

**Description**

Returns a clone, also known as a deep copy, of a variable. There is no reference to the original variable.

**Returns**

A clone of a variable.

**Category**

Structure functions, System functions

**Function syntax**

```
Duplicate(variable_name)
```

**See also**

StructCopy, other Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

**History**

ColdFusion 8: Changed behavior: this function can duplicate CFCs.

ColdFusion MX: Changed behavior: this function can be used on XML objects.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable_name</td>
<td>Name of a variable to duplicate</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to duplicate complex structures, such as nested structures and queries. When you duplicate a CFC instance, the entire CFC contents is copied, including the values of the variables in the this scope at the time you call the Duplicate function. Thereafter, the two CFC instances are independent, and changes to one copy, for example by calling one of its functions, have no effect on the other copy.

⚠️ **Note**

With this function, you cannot duplicate a COM, CORBA, or JAVA object returned from the cfobj ect tag or the CreateObject function. If an array element or structure field is a COM, CORBA, or JAVA object, you cannot duplicate the array or structure.

**Example**
<h3>Duplicate Example</h3>
<cfset s1 = StructNew()>
<cfset s1.nested = StructNew()>
<cfset s1.nested.item = "original">
<cfset copy = StructCopy(s1)>
<cfset clone = Duplicate(s1)>
<!--- modify the original --->
<cfset s1.nested.item = "modified">
<cfoutput>
<p>The copy contains the modified value: #copy.nested.item#</p>
<p>The duplicate contains the original value: #clone.nested.item#</p>
</cfoutput>
**Deserialize**

**Description**

Deserializes a String

**Returns**

A string

**Function syntax**

```
deserialize( String StringToBeDeserialized, String type, boolean useCustomSerializer )
```

**History**

ColdFusion 11: Added this function. For more information see [REST Enhancements in ColdFusion 11](#).

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringToBeDeserialized</td>
<td>A string that needs to be deserialized.</td>
</tr>
<tr>
<td>type</td>
<td>String. The type of the data to be deserialized. ColdFusion by default supports XML and JSON formats. You can also implement support for other types in the CustomSerializer CFC. See <a href="#">REST Enhancements in ColdFusion 11</a>.</td>
</tr>
<tr>
<td>useCustomSerializer</td>
<td>Boolean. Whether to use the custom serializer or not. The default value is true. The custom serializer will be always used for deserialization. If false, the XML/JSON deserialization will be done using the default ColdFusion behavior. If any other type is passed with useCustomSerializer as false, then TypeNotSupportedException will be thrown.</td>
</tr>
</tbody>
</table>

For more information on usage, see [this document](#).
### DeserializeXML

**Description**

Deserializes a string in XML format to a ColdFusion object.

**Returns**

Object

**Function syntax**

```plaintext
deserializeXML( String strToDeserialize, boolean useCustomSerializer )
```

**History**

ColdFusion 11: Added this function. For more information see REST Enhancements in ColdFusion 11.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringToBeDeserialized</td>
<td>Required. A string that needs to be deserialized.</td>
</tr>
<tr>
<td>useCustomSerializer</td>
<td>Boolean. Whether to use the custom serializer or not. The default value is</td>
</tr>
<tr>
<td></td>
<td>true. The custom serializer will be always used for XML deserialization. If</td>
</tr>
<tr>
<td></td>
<td>false, the XML/JSON deserialization will be done using the default ColdFu</td>
</tr>
<tr>
<td></td>
<td>sion behavior. If any other type is passed with useCustomSerializer as fa</td>
</tr>
<tr>
<td></td>
<td>lse, then TypeNotSupportedException will be thrown.</td>
</tr>
</tbody>
</table>

For more information on usage, see this document.
Functions e-g

- EncodeForCSS
- EncodeForHTML
- EncodeForHTMLAttribute
- EncodeForJavaScript
- EncodeForURL
- EncodeForXML
- EncodeForXMLAttribute
- EncodeForXpath
- Encrypt
- EncryptBinary
- EntityDelete
- EntityLoad
- EntityLoadByExample
- EntityLoadByPK
- EntityMerge
- EntityNew
- EntityReload
- EntitySave
- EntityToQuery
- Evaluate
- Exp
- ExpandPath
- FileClose
- FileCopy
- FileDelete
- FileExists
- FileGetMimeType
- FileIsEOF
- FileMove
- FileOpen
- FileRead
- FileReadBinary
- FileReadLine
- FileSeek
- FileSetAccessMode
- FileSetAttribute
- FileSetLastModified
- FileSkipBytes
- FileUpload
- FileUploadAll
- FileWrite
- FileWriteLine
- Find
- FindNoCase
- FindOneOf
- FirstDayOfMonth
- Fix
- FormatBaseN
- GeneratePBKDFKey
- GenerateSecretKey
- GetApplicationMetadata
- GetAuthUser
- GetBaseTagData
- GetBaseTagList
- GetBaseTemplatePath
- GetClientVariablesList
- GetComponentMetaData
- GetContextRoot
- GetCPUUsage
- GetCurrentTemplatePath
- GetDirectoryFromPath
- GetEncoding
- GetException
- GetFileFromPath
- GetFileInfo
- GetFunctionCalledName
- GetFunctionList
- GetGatewayHelper
- GetHTTPHeader
- GetHTTPHeaderData
- GetHTTPTimeString
- GetK2ServerDocCount
- GetK2ServerDocCountLimit
- GetLocale
- GetLocaleDisplayName
- GetLocalHostIP
- GetMetaData
- GetMetricData
- GetPageContext
- GetPrinterInfo
- GetPrinterList
- GetProfileSections
- GetProfileString
- GetReadableImageFormats
- GetSafeHTML
- GetSOAPRequest
- GetSOAPRequestHeader
- GetSOAPResponse
- GetSOAPResponseHeader
- GetSystemFreeMemory
- GetSystemTotalMemory
- GetTempDirectory
- GetTempFile
- GetTemplatePath
- GetTickCount
- GetTimeZoneInfo
- GetToken
- GetTotalSpace
- GetUserRoles
- GetVFSMetaData
- GetWriteableImageFormats
EncodeForCSS

Description

Encodes the input string for use in CSS.

Returns

Encoded string

Category

Display and formatting functions

Syntax

encodeForCSS(inputString [,canonicalize])

See also

EncodeForHTML, EncodeForHTMLAttribute, EncodeForURL, Canonicalize

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to encode.</td>
</tr>
<tr>
<td>canonicalize</td>
<td>Optional. If set to true, canonicalization happens before encoding. If set to false, the given input string will just be encoded. The default value for canonicalize is false. When this parameter is not specified, canonicalization will not happen. By default, when canonicalization is performed, both mixed and multiple encodings will be allowed. To use any other combinations you should canonicalize using canonicalize method and then do encoding.</td>
</tr>
</tbody>
</table>
<cfif not isDefined ("form.bgcolor")>
  <cfset form.bgcolor = 'red'>
</cfif>
<cfoutput>
<style>
.myDiv
{
  background-color : #encodeForCSS(form.bgcolor)#;
/* Encode the input to avoid any malicious code execution.*/
}
</style>
</cfoutput>
<hr/>
<cfoutput>
<div class="myDiv">
  This div element is styled!!!!
</div>
</cfoutput>
<hr/>
<cfform action="#cgi.SCRIPT_NAME#" method="post" >
  Background Color : <cfinput name="bgcolor" type="text" value="#form.bgcolor#">
  <br/>    <cfinput name="submit" type="submit" value="Style the div!!!"> 
</cfform>
EncodeForHTML

Description

Encodes the input string for use in HTML.

Returns

Encoded string

Category

Display and formatting functions

Syntax

`encodeForHTML(inputString [,canonicalize])`

See also

Canonicalize, EncodeForHTMLAttribute, EncodeForJavaScript, EncodeForCSS, EncodeForURL

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>inputString</code></td>
<td>Required. The string to encode.</td>
</tr>
<tr>
<td><code>canonicalize</code></td>
<td>Optional. If set to true, canonicalization happens before encoding. If set to false, the given input string will just be encoded. The default value for canonicalize is false. When this parameter is not specified, canonicalization will not happen. By default, when canonicalization is performed, both mixed and multiple encodings will be allowed. To use any other combinations you should canonicalize using canonicalize method and then do encoding.</td>
</tr>
</tbody>
</table>

Example
<cfif isDefined("form.submit")>
    <b>
    Output:<cfoutput >&gt;encodeForHTML(form.userName)&lt;/cfoutput>
    </b>
</cfif>
<cfelse>
    <cfset form.username=""/>
</cfif>
<cfform>
    <cfinput name="userName" type="text" value="#form.userName#">
    <cfinput name="submit" type="submit" value="Submit">
</cfform>
EncodeForHTMLAttribute

Description

Encodes the input string for use in HTML attribute, such as table width or image height.

Returns

Encoded string

Category

Display and formatting functions

Syntax

encodeForHTMLAttribute(inputString [,canonicalize])

See also

Canonicalize, EncodeForJavaScript, EncodeForCSS, EncodeForURL, EncodeForHTML

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to encode.</td>
</tr>
<tr>
<td>canonicalize</td>
<td>Optional. If set to true, canonicalization happens before encoding. If set to false, the given input string will just be encoded. The default value for canonicalize is false. When this parameter is not specified, canonicalization will not happen. By default, when canonicalization is performed, both mixed and multiple encodings will be allowed. To use any other combinations you should canonicalize using canonicalize method and then do encoding.</td>
</tr>
</tbody>
</table>

Example
<cfif not isDefined("form.submit")>
  <cfset form.width = 200 />
</cfif>
<cfoutput>
  <table width="#encodeForHTMLAttribute(form.width)#" border="1" bgcolor="RED">
    <tr>
      <td>
        Enter the value in the below text field.
      </td>
    </tr>
  </table>
</cfoutput>
<cfform>
  <cfinput name="width" type="text" value="#form.width#">
  <cfinput name="submit" type="submit" value="Submit">
</cfform>
EncodeForJavaScript

Description

Encodes the input string for use in JavaScript.

Returns

Encoded string

Category

Display and formatting functions

Syntax

encodeForJavaScript(inputString [,canonicalize])

See also

Canonicalize, EncodeForHTMLAttribute, EncodeForHTML, EncodeForCSS, EncodeForURL

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to encode.</td>
</tr>
<tr>
<td>canonicalize</td>
<td>Optional. If set to true, canonicalization happens before encoding. If set to false, the given input string will just be encoded. The default value for canonicalize is false. When this parameter is not specified, canonicalization will not happen. By default, when canonicalization is performed, both mixed and multiple encodings will be allowed. To use any other combinations you should canonicalize using canonicalize method and then do encoding.</td>
</tr>
</tbody>
</table>

Example
<cfif isDefined("form.submit")>
<!--- If the user submits the form, show him/her a JavaScript alert. --->
<cfoutput>
<script type="text/javascript">
  alert(’Hello #encodeForJavascript(form.userName)# !!!’);
  // For security purpose, encode the user-generated input, so that it does not execute malicious codes.
</script>
</cfoutput>
<cfelse>
  <cfset form.username = "" />
</cfif>
</cifform>

<cfinput name="userName" type="text" value="#form.userName#">
<cfinput name="submit" type="submit" value="SayHello!!!">
EncodeForURL

Description

Encodes the input string for use in URLs.

Returns

Encoded string

Category

Display and formatting functions

Syntax

encodeForURL(inputString [,canonicalize])

See also

Canonicalize, EncodeForHTMLAttribute, EncodeForHTML, EncodeForCSS, EncodeForJavaScript

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to encode.</td>
</tr>
<tr>
<td>canonicalize</td>
<td>Optional. If set to true, canonicalization happens before encoding. If set to false, the given input string will just be encoded. The default value for canonicalize is false. When this parameter is not specified, canonicalization will not happen. By default, when canonicalization is performed, both mixed and multiple encodings will be allowed. To use any other combinations you should canonicalize using canonicalize method and then do encoding.</td>
</tr>
</tbody>
</table>

Example
<cfif not isDefined ("form.url")>
    <cfset form.url = "www.adobe.com">
</cfif>
<cfform action="#cgi.SCRIPT_NAME#" method="post">
    <cfinput name="url" type="text" value="#form.url#">
    <cfinput name="submit" type="submit" value="Show link to this URL!!!">
</cfform>
<hr />
<cfoutput>
    <b>LINK to URL:</b> <a target="_blank"
        href="http://#encodeForURL(form.url)#">#encodeForURL(form.url)#</a>
</cfoutput>
EncodeForXML

Description

Encodes a string for XML.

Returns

Encoded XML string.

Category

Display and formatting functions.

Function syntax

```
encodeforXML(Inputstring , [canonicalize])
```

See also

Canonicalize, EncodeForHTMLAttribute, EncodeForJavaScript, EncodeForCSS, EncodeForURL

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to encode.</td>
</tr>
<tr>
<td>canonicalize</td>
<td>Optional. If set to true, canonicalization happens before encoding. If set to false, the given input string will just be encoded. The default value for canonicalize is false. When this parameter is not specified, canonicalization will not happen. By default, when canonicalization is performed, both mixed and multiple encodings will be allowed. To use any other combinations you should canonicalize using canonicalize method and then do encoding.</td>
</tr>
</tbody>
</table>

Example
<cfif isDefined("form.submit")>
    MyDoc = XmlNew();
    MyDoc.xmlRoot = XmlElemNew(MyDoc,"MyRoot");
    MyDoc.MyRoot.XmlText = #encodeForXML(form.userName)#;
    <b>
    Output:<cfoutput >#encodeForXML(form.userName)#</cfoutput>
    </b>
</cfelse>
<cfset form.username=""/>
</cfif>
<cfform>
    <cfinput name="userName" type="text" value="#form.userName#">
    <cfinput name="submit" type="submit" value="Submit">
</cfform>
Encrypt

Description

Encrypts a string using a specific algorithm and encoding method.

Returns

String; can be much longer than the original string.

Category

Security functions, String functions

Function syntax

Encrypt(string, key [, algorithm, encoding, IVorSalt, iterations])

See also

Decrypt, EncryptBinary, DecryptBinary

History

ColdFusion 8: Added support for encryption using the RSA BSafe Crypto-J library on Enterprise Edition.

ColdFusion MX 7.01: Added the IVorSalt and iterations parameters.

ColdFusion MX 7: Added the algorithm and encoding parameters.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>String to encrypt.</td>
</tr>
<tr>
<td>key</td>
<td>String. Key or seed used to encrypt the string.</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>• For the CFMX.COMPAT algorithm, any combination of any number of characters; used as a seed used to generate a 32-bit encryption key.</td>
</tr>
<tr>
<td></td>
<td>• For all other algorithms, a key in the format used by the algorithm. For these algorithms, use the <code>generateSecretKey</code> function to generate the key.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>algorithm</th>
<th>(Optional) The algorithm to use to encrypt the string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The Enterprise Edition of ColdFusion installs the RSA BSafe Crypto-J library, which provides FIPS-140 Compliant Strong Cryptography. It includes the following algorithms:</td>
</tr>
<tr>
<td></td>
<td>• AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197.</td>
</tr>
<tr>
<td></td>
<td>• DES: the Data Encryption Standard algorithm defined by NIST FIPS-46-3.</td>
</tr>
<tr>
<td></td>
<td>• DES-EDE: the &quot;Triple DES&quot; algorithm defined by NIST FIPS-46-3.</td>
</tr>
<tr>
<td></td>
<td>• DESX: The extended Data Encryption Standard symmetric encryption algorithm.</td>
</tr>
<tr>
<td></td>
<td>• RC2: The RC2 block symmetric encryption algorithm defined by RFC 2268.</td>
</tr>
<tr>
<td></td>
<td>• RC4: The RC4 symmetric encryption algorithm.</td>
</tr>
<tr>
<td></td>
<td>• RC5: The RC5 encryption algorithm.</td>
</tr>
<tr>
<td></td>
<td>• PBE: Password-based encryption algorithm defined in PKCS #5.</td>
</tr>
</tbody>
</table>

| In addition to these algorithms, you can use the algorithms provided in the Standard Edition of ColdFusion. |

© 2014 Adobe Systems Incorporated. All rights reserved.
The Standard Edition of ColdFusion installs a cryptography library with the following algorithms:

- **CFMX_COMPAT**: the algorithm used in ColdFusion MX and prior releases. This algorithm is the least secure option (default).
- **AES**: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197.
- **BLOWFISH**: the Blowfish algorithm defined by Bruce Schneier.
- **DES**: the Data Encryption Standard algorithm defined by NIST FIPS-46-3.
- **DESEDE**: the "Triple DES" algorithm defined by NIST FIPS-46-3.

If you install a security provider with additional cryptography algorithms, you can also specify any of its string encryption and decryption algorithms.

<table>
<thead>
<tr>
<th><strong>encoding</strong></th>
<th>(Optional; if you specify this parameter, also specify the algorithm parameter). The binary encoding in which to represent the data as a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Base64: the Base64 algorithm, as specified by IETF RFC 2045.</td>
</tr>
<tr>
<td></td>
<td>• Hex: the characters A-F0-9 represent the hexadecimal byte values.</td>
</tr>
<tr>
<td></td>
<td>• UU: the UUEncode algorithm (default).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>IVorSalt</strong></th>
<th>(Optional) Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, also specify the <em>algorithm</em> parameter.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• For Block Encryption algorithms: This is the binary Initialization Vector value to use with the algorithm. The algorithm must contain a Feedback Mode other than ECB. This must be a binary value that is exactly the same size as the algorithm block size. Use the same value in the <em>decrypt</em> function to successfully decrypt the data.</td>
</tr>
<tr>
<td></td>
<td>• For Password Based Encryption algorithms: This is the binary Salt value to transform the password into a key. The same value must be used to decrypt the data.</td>
</tr>
</tbody>
</table>
(Optional) The number of iterations to transform the password into a binary key. Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, also specify the algorithm parameter with a Password Based Encryption (PBE) algorithm. Do not specify this parameter for Block Encryption algorithms. Use the same value to encrypt and decrypt the data.

Usage

This function uses a symmetric key-based algorithm, in which the same key is used to encrypt and decrypt a string. The security of the encrypted string depends on maintaining the secrecy of the key.

The following are the FIPS-140 approved algorithms included in the RSA BSafe Crypto-J library that are used by ColdFusion. Some of these are not used with the encrypt function, but are used with other functions:

- AES – ECB, CBC, CFB (128), OFB (128) – [128, 192, 256-bit key sizes]
- AES – CTR
- Diffie-Hellman Key Agreement
- DSA
- FIPS 186-2 General Purpose [(x-Change Notice); (SHA-1)]
- FIPS 186-2 [(x-Change Notice); (SHA-1)]
- HMAC-SHAx (where x is 1, 224, 256, 384, or 512)
- RSA PKCS#1 v1.5 (sign, verify) (SHA-1,SHA-224,SHA-256,SHA-384,SHA-512)
- Secure Hash Standard (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)
- Triple DES - ECB, CBC, CFB (64 bit), and OFB (64 bit)

All algorithms included in the RSA BSafe Crypto-J library are available for use in the Enterprise Edition. In certain cases, you may want to disable some algorithms. To disable the DESX, RC5, and MD5PRNG algorithms, specify the following in the JVM arguments on the Java and JVM page of the ColdFusion Administrator:

```
-Dcoldfusion.enablefipscrypto=true
```

FIPS-140 approved cryptography is not available if you are running ColdFusion on WebSphere of JBoss.

To use the IBM/Lotus Sametime Instant Messaging Gateway in the Enterprise edition, disable the FIPS-140-only cryptography setting by specifying the following in the JVM arguments on the Java and JVM page of the ColdFusion Administrator:
In Standard Edition, for all algorithms except the default algorithm, ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, Adobe cannot provide technical support for third-party security providers.

The default algorithm, which is the same one used in ColdFusion 5 and ColdFusion MX, uses an XOR-based algorithm that uses a pseudo-random 32-bit key, based on a seed passed by the user as a function parameter. This algorithm is less secure than the other available algorithms.

Example

The following example encrypts and decrypts a text string. It lets you specify the encryption algorithm and encoding technique. It also has a field for a key seed to use with the CFMX_COMPAT algorithm. For all other algorithms, it generates a secret key.
Encrypt Example

<!---- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>
  <cfscript>
    /* GenerateSecretKey does not generate key for the CFMX_COMPAT algorithm, so use the key from the form. */
    if (Form.myAlgorithm EQ "CFMX_COMPAT")
      theKey=Form.MyKey;
    // For all other encryption techniques, generate a secret key.
    else
      theKey=generateSecretKey(Form.myAlgorithm);
    //Encrypt the string
    encrypted=encrypt(Form.myString, theKey, Form.myAlgorithm, Form.myEncoding);
    //Decrypt it
    decrypted=decrypt(encrypted, theKey, Form.myAlgorithm, Form.myEncoding);
  </cfscript>

<!---- Display the values used for encryption and decryption, and the results. --->
<cfoutput>
<b>The algorithm:</b> #Form.myAlgorithm#<br>
<b>The key:</b> #theKey#<br>
<b>The string:</b> #Form.myString# <br>
<b>Encrypted:</b> #encrypted#<br>
<b>Decrypted:</b> #decrypted#<br>
</cfoutput>
</cfif>

<!---- The input form.---->
<form action="#CGI.SCRIPT_NAME#" method="post">
<b>Select the encoding</b><br>
<select size="1" name="myEncoding">
  <option selected>UU</option>
  <option>Base64</option>
  <option>Hex</option>
</select><br>
<b>Select the algorithm</b><br>
<select size="1" name="myAlgorithm">
  <option selected>CFMX_COMPAT</option>
  <option>AES</option>
  <option>DES</option>
  <option>DESEDE</option>
</select><br>
<b>Input your key</b> (used for CFMX_COMPAT encryption only)<br>
<input type = "Text" name = "myKey" value = "MyKey"><br>
<b>Enter string to encrypt</b><br>
<textarea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">This string will be encrypted (you can replace it with more typing). </textarea><br>
<input type = "Submit" value = "Encrypt my String">
</form>
GetTempDirectory

Description

Gets the path of the directory that ColdFusion uses for temporary files. The directory depends on the account under which ColdFusion is running and other factors. Before using this function in an application, test to determine the directory it returns under your account.

Returns

The absolute pathname of a directory, including a trailing slash, as a string.

Category

System functions

Function syntax

GetTempDirectory()

See also

GetTempFile

History

ColdFusion MX: Changed behavior: on Windows, this function now returns the temporary directory of the embedded Java application server. On other platforms, it returns the temporary directory of the operating system.

Example

```xml
<h3>GetTempDirectory Example</h3>
<p>The temporary directory for this ColdFusion server is</p>
<cfoutput>#GetTempDirectory()#</cfoutput>.</p>
<p>We have created a temporary file called</p>
<cfoutput>#GetTempFile(GetTempDirectory(),"testFile")#</cfoutput></p>
```
GetApplicationMetadata

Description

Returns the application settings that you have specified in the application, either in the Application.cfc or Application.cfm. For details, see Application Variables section in CFML Reference.

⚠️ If you have turned on Enable Global Script Protection in ColdFusion Administrator (Server Settings > Settings), the value returned is the default scopes protected (form, URL, CGI, or Cookie). If you specify an invalid value, none, or if Enable Global Script Protection is turned off in ColdFusion Administrator, none is returned.

⚠️ If you have specified memory limit for VFS in ColdFusion Administrator and if Memory Limit per Application for In-Memory Virtual File System (Server Settings > Settings) has a value lesser than what you have specified in the Application.cfc (this.inmemoryfilesystem.size), then the returned value is the one specified in the ColdFusion Administrator and not the Application.cfc.

⚠️ If ColdFusion does not find the Application.cfc/Application.cfm ColdFusion searches for the application in the order in which you have set Application.cfm/Application.cfc look up order (ColdFusion Administrator > Settings > Server Settings). If no application is found, an empty struct is returned.

Returns

A struct that contains application settings such as name, sessionManagement, or invokeImplicitAccessor.

Category

Other functions

Syntax

getApplicationMetadata()

Example

The example shows how you can access application settings in the CFM using the function getApplicationMetadata:

Application.cfc
component
{
    this.name = "myapp";
    this.applicationtimeout = createtimespan(0, 0, 0, 10);
    this.sessiontimeout = createtimespan(0, 0, 0, 10);
    this.sessionmanagement = true;
    this.clientmanagement = true;
    this.datasource = "cfartgallery";
    this.ormenabled = true;
    this.secureJSON = true;
    this.secureJSONPrefix = "///";
    this.setClientCookies = true;
    this.setDomainCookies = true;
    this.setClientStorage = "Registry";
    this.setLoginStorage = "Cookies";
    this.scriptProtect = "all";
    this.mappings["mymapping"] = getdirectoryfrompath(cgi.cf_template_path);
    this.customTagPaths = "path1,path2,path3";
    this.invokeImplicitAccessor = true;
    this.inmemoryfilesystem.enabled = true;
    this.inmemoryfilesystem.size = 10;
}

AppMetaData.cfm

<cfscript>
    writedump(getApplicationMetadata());
</cfscript>
EncryptBinary

Description

Encrypts binary data using a specific algorithm and encoding method.

Returns

Binary data.

Category

Security functions, String functions

Function syntax

EncryptBinary(bytes, key [, algorithm, IVorSalt, iterations])

See also

Decrypt, DecryptBinary, Encrypt

History

ColdFusion 8: Added support for encryption using the RSA BSafe Crypto-J library on Enterprise Edition.
ColdFusion MX 7.01: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bytes</td>
<td>Binary data to encrypt.</td>
</tr>
</tbody>
</table>
| key       | String. Key or seed used to encrypt the string.  
  - For the CFXM_COMPAT algorithm, any combination of any number of characters; used as a seed used to generate a 32-bit encryption key.  
  - For all other algorithms, a key in the format used by the algorithm. For these algorithms, use the GenerateSecretKey function to generate the key. |
### algorithm

(Optional) The algorithm to use to decrypt the string. The Enterprise Edition of ColdFusion installs the RSA BSafe Crypto-J library, which provides FIPS-140 Compliant Strong Cryptography. For a list of algorithms, see the `Encrypt` function. The Standard Edition of ColdFusion installs a cryptography library with the following algorithms:

- CFMX_COMPAT: the algorithm used in ColdFusion MX and prior releases. This algorithm is the least secure option (default).
- AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197.
- BLOWFISH: the Blowfish algorithm defined by Bruce Schneier.
- DESEDE: the "Triple DES" algorithm defined by NIST FIPS-46-3.

If you install a security provider with additional cryptography algorithms, you can also specify any of its string encryption and decryption algorithms.

### IVorSalt

(Optional) Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, also specify the `algorithm` parameter.

- For Block Encryption algorithms: This is the binary Initialization Vector value to use with the algorithm. The algorithm must contain a Feedback Mode other than ECB. This must be a binary value that is exactly the same size as the algorithm block size. Use the same value in the `Decrypt` function to successfully decrypt the data.
- For Password Based Encryption algorithms: This is the binary Salt value to transform the password into a key. The same value must be used to decrypt the data.

### iterations

(Optional) The number of iterations to transform the password into a binary key. Specify this parameter to adjust ColdFusion encryption to match the details of other encryption software. If you specify this parameter, also specify the `algorithm` parameter with a Password Based Encryption (PBE) algorithm. Do not specify this parameter for Block Encryption algorithms. Use the same value to encrypt and decrypt the data.

### Usage
This function uses a symmetric key-based algorithm, in which the same key is used to encrypt and decrypt binary data. The security of the encrypted data depends on maintaining the secrecy of the key. For all algorithms except the default algorithm, ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, Adobe cannot provide technical support for third-party security providers. The default algorithm, which is the same as was used in ColdFusion 5 and ColdFusion MX, uses an XOR-based algorithm that uses a pseudo-random 32-bit key, based on a seed passed by the user as a function parameter. This algorithm is less secure than the other available algorithms.

Example

The following example encrypts and decrypts binary data. It encrypts the binary data contained in a file and then decrypts the encrypted file. It lets you specify the encryption algorithm and encoding technique. It also has a field for a key seed to use with the CFMX_COMPAT algorithm. For all other algorithms, it generates a secret key.
<h3>EncryptBinary Example</h3>

<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myfile")>

<cfile file="#Form.myfile#" action="readBinary" variable="myData">
<cfscript>
/* GenerateSecretKey does not generate key for the CFMX_COMPAT algorithm, so use the key from the form. */
if (Form.myAlgorithm EQ "CFMX_COMPAT")
theKey=Form.MyKey;
// For all other encryption techniques, generate a secret key.
else
theKey=generateSecretKey(Form.myAlgorithm);
//Encrypt the string
encrypted=encryptBinary(myData, theKey, Form.myAlgorithm);
//Decrypt it
decrypted=decryptBinary(encrypted, theKey, Form.myAlgorithm);
</cfscript>
<cfset encfile="#Form.myfile#" & ":enc">
<cfset decfile="#Form.myfile#" & ":dec">
<cfile file="#encfile#" action="write" output="#encrypted#">
<cfile file="#decfile#" action="write" output="#decrypted#">

<!--- Display the values used for encryption and decryption, and the results. --->
<cfoutput>
<br>The algorithm:<b> "#Form.myAlgorithm#"</b>
<br>The key:<b> "#theKey#"</b>
<br>The InputFile:<b> "#Form.myfile#"</b>
<br>Encrypted:<b> "#encfile#"</b>
<br>Decrypted:<b> "#decfile#"</b>
</cfoutput>
</cfif>

<!--- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
<br>Select the algorithm<br>
<select size="1" name="myAlgorithm">
<option selected>CFMX_COMPAT</option>
<option>AES</option>
<option>DES</option>
<option>DESEDE</option>
</select><br>
<br>Input your key (used for CFMX_COMPAT encryption only)<br>
<input type = "Text" name = "myKey" value = "MyKey"><br>
<br>Enter filename to encrypt<br>
<input type="text" name="myfile" value="Enter the path of the file to encrypt"><br>
<input type = "Submit" value = "Encrypt file ">
</form>
**EntityDelete**

**Description**

Deletes the record from the database for the specified entity. Depending on the cascade attribute specified in the mapping, it deletes the associated objects also.

**Category**

ORM functions

**Function Syntax**

```
EntityDelete(entity)
```

**See Also**

`EntityLoad`, `EntitySave`

**History**

ColdFusion 9: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity</td>
<td>Name of the entity being deleted.</td>
</tr>
</tbody>
</table>

**Example**

```
<cfset employee = EntityLoad('employee', 100, true)>
<cfset EntityDelete(employee)>
<cfset employee = CreateObject('component', 'employee')>
<cfset employee.setEmployeeID(100)>
<cfset EntityDelete(employee)>
```
**EntityLoad**

**Description**

Loads and returns an array of entities for the specified entity name. You can also specify a filter criteria and sort order. All EntityLoad methods take the entity name as input.

**Returns**

Array (if unique=false) Single entity (if unique=true)

**Category**

ORM functions

**Function syntax**

```
EntityLoad (entityname)
EntityLoad (entityname, id [, unique])
EntityLoad (entityname, filtercriteria [,unique]
EntityLoad(entityname, filtercriteria, sortorder [, options])
```

**See Also**

EntityLoadByExample, EntityReload, EntityDelete

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity name</td>
<td>Name of the entity to be loaded.</td>
</tr>
<tr>
<td>ID</td>
<td>The primary key value of the entity that must be loaded. If the entity has a composite key, then the id has to be specified as key-value pairs (ColdFusion struct).</td>
</tr>
<tr>
<td>unique</td>
<td>If unique is set to true, then the entity is returned. If you are sure that only one record exists that matches this filtercriteria, then you can specify unique=true, so that a single entity is returned instead of an array. If you set unique=true and multiple records are returned, then an exception occurs.</td>
</tr>
<tr>
<td>filtercriteria</td>
<td>Key-value pair (ColdFusion Struct) of property names and values. If there are more than one key-value pair, then the AND operator is used. If specified, loads and returns an array of entities of the specified entity name that matches the filtercriteria.</td>
</tr>
<tr>
<td><strong>sortorder</strong></td>
<td>String used to specify the sort order of the entities that are returned. If specified, loads and returns an array of entities that satisfy the filter criteria sorted as specified by the sort order.</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>options</strong></td>
<td>The following options to customize the output:</td>
</tr>
<tr>
<td></td>
<td>• ignorecase: Ignores the case of sort order when set to true. Use only if you specify the sortorder parameter.</td>
</tr>
<tr>
<td></td>
<td>• offset: Specifies the position from which to retrieve the objects.</td>
</tr>
<tr>
<td></td>
<td>• maxResults: Specifies the maximum number of objects to be retrieved.</td>
</tr>
<tr>
<td></td>
<td>• cacheable: Whether the result has to be cached in the secondary cache. Default is false.</td>
</tr>
<tr>
<td></td>
<td>• cachename: Name of the cache in secondary cache.</td>
</tr>
<tr>
<td></td>
<td>• timeout: Specifies the timeout value (in seconds) for the query.</td>
</tr>
</tbody>
</table>

**History**

ColdFusion 9: Added this function.

**Usage**

For pagination, you can use the options offset and maxResults as shown in the example:

```coldfusion
EntityLoad('employee', {department='qa'}, {offset=21, maxResults=10})
```

This example retrieves the (next) 10 objects of employees whose department is qa from offset 22.

**Example**

Example with only entity name specified:

```coldfusion
<cfset employees = EntityLoad('employee')>
```

Example with EntityName, id, and unique set to true. Instead of true, if you set unique as false, then array is returned with one entity.

```coldfusion
<cfset employee = EntityLoad('employee', 100, true)>
<cfset orderDetail = EntityLoad('orderdetails', {OrderID=100, ProductID=1}, true)>
```
Example which describes how to retrieve objects whose country is UK, and sorted by Department ascending and Age descending:

```cfml
<cfset employeesInUKSorted = EntityLoad('employee', 
{country="UK"}, "Department Asc, Age Desc")>
```

Example that describes how to retrieve details of all the employees who live in 'UK':

```cfml
<cfset employeesFromUK = EntityLoad('employee', {country="UK"})>
```

Example that describes how to retrieve a unique object. If you specify `unique= "true"` and more than one object satisfies the condition, then an exception occurs.

```cfml
<cfset employee = EntityLoad('employee', {firstname="Marcia", lastname="Em"}, 
"true")>
```
EntityLoadByExample

Description

Loads and returns an array of objects that match the sampleentity. The filter criteria is constructed by ANDing all the non-null properties of the sampleentity.

Returns

Array of objects

Category

ORM functions

Function Syntax

entityloadbyexample(sampleentity [, unique])

See Also

EntityLoad, EntityReload

History

ColdFusion 9: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleentity</td>
<td>Name of the sample entity that is used to match and filter similar entities to load.</td>
</tr>
</tbody>
</table>

Example

<cfset employee= CreateObject("component", "employee")>
<cfset employee.setDepartment("ColdFusion")>
<cfset employee.setCountry("USA")>
<cfset employee=EntityLoadByExample(employee)>
**EntityLoadByPK**

**Description**

 Loads and returns an array of objects for the specified primary key. Use this function to avoid specifying the boolean parameter that you must specify with the `EntityLoad()` function.

**Returns**

 object

**Category**

 ORM functions

**Function Syntax**

```plaintext
entityLoadByPK( entityName ,id)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity name</td>
<td>Name of the entity to be loaded.</td>
</tr>
<tr>
<td>id</td>
<td>Primary key id</td>
</tr>
</tbody>
</table>

**See Also**

 `EntityLoad`, `EntityReload`, `EntityLoadByExample`, `EntityDelete`, `ColdFusion ORM` chapter in *Developing ColdFusion Applications*

**History**

ColdFusion 9: Added this function.

**Example**

```cfscript
<cfscript>
art = EntityLoadByPK("Art", 1);
writedump(art);
</cfscript>
```
**EntityMerge**

**Description**
Attaches the specified entity to the current ORM session. It copies the state of the given object onto the persistent object with the same identifier and returns the persistent object. If there is no persistent instance currently associated with the session, it is loaded. The given instance is not associated with the session. User have to use the returned object from this session.

**Returns**
object

**Category**
ORM functions

**Function Syntax**

```
entityMerge( entity)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity</td>
<td>The entity that must be attached to the ORM session.</td>
</tr>
</tbody>
</table>

**See Also**
EntityLoad, EntityLoadByExample, EntityDelete, ColdFusion ORM chapter in Developing ColdFusion Applications

**History**
ColdFusion 9: Added this function.
**EntityNew**

⚠️ **Note**

This function has been enhanced in ColdFusion 9.0.1 to support multiple data sources in the same application. Install the ColdFusion 9 update to use the feature.

**Description**

Creates an instance of the persistent CFC with the entity name that you provide.

**Returns**

Object

**Category**

ORM functions

**Function Syntax**

`entityNew(entityName [ ,properties])`

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>entityName</code></td>
<td>Entity name of the persistent CFC.</td>
</tr>
<tr>
<td><code>properties</code></td>
<td>Key-value pair (ColdFusion struct) of property names and values.</td>
</tr>
</tbody>
</table>

**See Also**

`EntityLoad`, `EntityLoadByExample`, `EntityDelete`, `ColdFusion ORM` chapter in *Developing ColdFusion Applications*

**History**

ColdFusion 9: Added this function.

**Usage**

The enhancement in ColdFusion 9.0.1 allows the application to initialize the object that is being created. `properties` takes a struct with key being the property name. When the object is created, all the properties are populated with the passed struct. For example,

```cfml
cfset artistObj = entityNew("Artists",{FirstName="Tom",LastName="Ron"})
```

**Example**
newArtistObj = EntityNew("Artists");
newArtistObj.setfirstname("John");
newArtistObj.setlastname("Smith");
newArtistObj.setaddress("5 Newport lane");
newArtistObj.setcity("San Francisco");
newArtistObj.setstate("CA");
newArtistObj.setPostalCode("90012");
newArtistObj.setphone("612-832-2343");
newArtistObj.setfax("612-832-2344");
newArtistObj.setemail("jsmith@company.com");
newArtistObj.setThePassword("jsmith");
EntitySave(newartistobj);
ormflush();
**EntityReload**

**Description**

Reloads data for an entity that is already loaded. This method refetches data from the database and repopulates the entity with the refreshed data.

**Category**

ORM functions

**Function Syntax**

```coldfusion
entityreload()
```

**See Also**

EntityLoad, EntityLoadByExample, EntityDelete

**History**

ColdFusion 9: Added this function.
**EntitySave**

**Description**

Saves or updates data of the entity and all related entities to the database. ColdFusion automatically tries to find if a new record should be inserted or an existing record be updated for the given entity. If you set `forceinsert=true`, then ColdFusion tries to insert the entity as a new record.

**Returns**

Void

**Category**

ORM functions

**Function Syntax**

```
EntitySave(entity, [forceinsert])
```

**See Also**

`EntityLoad`, `EntityLoadByExample`, `EntityDelete`

**History**

ColdFusion 9: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity</td>
<td>Name of the entity that must be saved in the database.</td>
</tr>
<tr>
<td>forceinsert</td>
<td>If true, then ColdFusion always tries to insert the entity as a new record.</td>
</tr>
</tbody>
</table>

**Example**

To save an entity:

```cfml
<cfset employee = createObject("Employee")>
<cfset employee.setFirstName("Marcia")>
<cfset employee.setlastName("Em")>
<cfset EntitySave(employee)>
```

To update an entity:
<cfset employee = EntityLoad('employee', 100, true)>
<cfset employee.setLastName("Em")>
<cfset EntitySave(employee)>
**EntityToQuery**

**Description**

Converts the input entity object or the input array of entity objects to a query object.

**Returns**

Query

**Category**

ORM functions

**Function Syntax**

```
EntityToQuery (orm_object, [entity_name])
EntityToQuery (orm_object_array, [entity_name])
```

**Parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orm_object</td>
<td>Entity object that needs to be converted to a query object.</td>
</tr>
<tr>
<td>orm_object_array</td>
<td>Array that needs to be converted to a query object.</td>
</tr>
<tr>
<td>entity_name</td>
<td>Name of the entity. Use this optional parameter to return the query of the given entity in the case of inheritance mapping.</td>
</tr>
</tbody>
</table>

**See Also**

EntityLoad, EntityDelete

**History**

ColdFusion 9: Added this function.

**Usage**

The following conditions apply for this function:

- In the case of array input, all objects in the array must be of the same type.
- The result query will not contain any relation data.

**Example**
<cfset artists = EntityLoad("Artist")>
<cfset artistQuery = EntityToQuery(artists)>
Evaluate

Description

Evaluates one or more string expressions, dynamically, from left to right. (The results of an evaluation on the left can have meaning in an expression to the right.) Returns the result of evaluating the rightmost expression.

Returns

An object; the result of the evaluations.

Category

Dynamic evaluation functions

Function syntax

Evaluate(string_expression1 [, string_expression2 , ... ])

See also

DE, IIF, PrecisionEvaluate, Using Expressions and Number Signs in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string_expression1, string_expression2...</td>
<td>Expressions to evaluate</td>
</tr>
</tbody>
</table>

Usage

String expressions can be complex. If a string expression contains a single- or double-quotiation mark, the mark must be escaped. This function is useful for forming one variable from multiple variables. For example, to reference a column of the query qNames with a variable, var, using an index value to traverse rows, you could use the following code:

```<cfset var=Evaluate("qNames.#colname#[#index#]")>```

Example
This example shows the use of PrecisionEvaluate and DE functions.

```
<h3>Evaluate Example</h3>
<cfif IsDefined("FORM.myExpression")>
  <cftry>
    <!--- Evaluate the expression --->
    <cfset theExpression = Evaluate(Form.myExpression)>
    <cfoutput>
      <!--- The DE function prevents the Evaluate function from evaluating
           the expression. --->
      The value of the expression #Evaluate(DE(FORM.MyExpression))#
      is #theExpression#.<br>
      <!--- The following line does not use the DE function. --->
      The value of the expression #FORM.MyExpression#
      is #theExpression#.<br>
    </cfoutput>
  </cftry>
  <cfcatch type="application">
    <cfoutput>Could not evaluate the expression #Form.myExpression#.<br>
    Make sure you enter a valid ColdFusion Expression.</cfoutput>
  </cfcatch>
</cfif>

<h3>Enter a ColdFusion expression for evaluation</h3>
<cfinput type="text" name="myExpression" size="60"><br />
<cfinput type="submit" name="submit">
```

© 2014 Adobe Systems Incorporated. All rights reserved. 1625
Exp

Description

Calculates the exponent whose base is e that represents \textit{number}. The constant e equals 2.71828182845904, the base of the natural logarithm. This function is the inverse of \textit{log}, the natural logarithm of \textit{number}.

Returns

The constant e, raised to the power of \textit{number}.

Category

\textbf{Mathematical functions}

Function syntax

\begin{verbatim}
Exp (number)
\end{verbatim}

See also

\textit{Log}, \textit{Log10}

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Exponent to apply to the base e</td>
</tr>
</tbody>
</table>

Usage

To calculate powers of other bases, use the exponentiation operator (^).

Example
<h3>Exp Example</h3>

<cfif IsDefined("FORM.Submit")>
<cfoutput>
<p>Your number, #FORM.number#</p>
<br>#FORM.number# raised to the E power: #exp(FORM.number)#
<cfif FORM.number LTE 0>
<br>You must enter a positive real number to see its natural logarithm
<cfelse><br>
The natural logarithm of #FORM.number#: #log(FORM.number)#
</cfif>
<cfif FORM.number LTE 0><br>
You must enter a positive real number to see its logarithm to base 10
<cfelse><br>
The logarithm of #FORM.number# to base 10: #log10(FORM.number)#
</cfif>
</cfoutput>
</cfif>
</cfoutput>
</cfif>
<cfform action = "exp.cfm">
Enter a number to see its value raised to the E power, its natural logarithm, and the logarithm of number to base 10.
<cfinput type = "Text" name = "number" message = "You must enter a number"
validate = "float" required = "No">
<input type = "Submit" name = "Submit">
</cfform>
ExpandPath

Description

Creates an absolute, platform-appropriate path that is equivalent to the value of `relative_path`, appended to the base path. This function (despite its name) can accept an absolute or relative path in the `relative_path` parameter. The base path is the currently executing page’s directory path. It is stored in `pageContext.getServletContext()`.

Returns

A string. If the relative path contains a trailing forward slash or backward slash, the return value contains the same trailing character.

Category

System functions

Function syntax

```
ExpandPath(relative_path)
```

See also

FileExists, GetCurrentTemplatePath, GetFileFromPath

History

ColdFusion MX: Changed behavior for the `relative_path` parameter: this function can now accept an absolute or relative path in the `relative_path` parameter. To resolve a path, this function uses virtual mappings that are defined in the ColdFusion Administrator. This function does not reliably use virtual mappings that are defined in IIS, Apache, or other web servers.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>relative_path</code></td>
<td>Relative or absolute directory reference or filename, within the current directory, (\ and ..) to convert to an absolute path. Can include forward or backward slashes. Files in the custom tag directory are also resolved. For instance, if there is a file test.txt in the custom tag directory (C:\Zeus), the function (with \test.txt) returns C:\Zeus\test.txt.</td>
</tr>
</tbody>
</table>

Usage

If the parameter or the returned path is invalid, the function throws an error. You cannot use this function with in-memory files. These examples show the valid constructions of `relative_path`:

- `ExpandPath( ".*" )`
- `ExpandPath( "/" )`
- `ExpandPath( "\" )`
ExpandPath( "/mycfpage.cfm")
ExpandPath( "mycfpage.cfm")
ExpandPath( "myDir/mycfpage.cfm")
ExpandPath( "/myDir/mycfpage.cfm")
ExpandPath( "/../mycfpage.cfm")

Enhancement in ColdFusion 10 lets files in the custom tag directory to also resolve. For instance, if there is a file test.txt in the custom tag directory (C:\Zeus), the function (with \test.txt) returns C:\Zeus\test.txt.

Example

```cfnl
<h3>ExpandPath Example</h3>
<cfset thisPath=ExpandPath("*.*")>
<cfset thisDirectory=GetDirectoryFromPath(thisPath)>
<cfoutput>
The current directory is: #GetDirectoryFromPath(thisPath)#
</cfoutput>
<cfif IsDefined("form.yourFile") AND form.yourFile is not ">
  <cfset yourFile = form.yourFile>
  <cfif FileExists(ExpandPath(yourfile))>
    <p>Your file exists in this directory. You entered the correct filename, #GetFileFromPath("#thisPath#/#yourfile#")#</p>
  <cfelse>
    <p>Your file was not found in this directory: Here is a list of the other files in this directory:</p>
    <!--- use cfdirectory to give the contents of the snippets directory, order by name and size --->
    <cfdirectory directory="#thisDirectory#" name="myDirectory" sort="name ASC, size DESC">
      <!--- Output the contents of the cfdirectory as a cftable --->
      <cftable query="myDirectory">
        <cfcol header="NAME:" text="#Name#">
        <cfcol header="SIZE:" text="#Size#">
      </cftable>
    </cfdirectory>
  </cfelse>
</cfif>
</cfoutput>
</cfif>
<form action="expandpath.cfm" method="post">
<h3>Enter the name of a file in this directory <i>(try expandpath.cfm)</i></h3>
<input type="text" name="yourFile">
<input type="submit" name="">
</form>
```
FileClose

Description

Closes an on-disk or in-memory file that is open. When you use the FileOpen function, ColdFusion returns a handle to a file object. When you close the file, the handle is still available; however, it lists the file as closed.

Category

System functions

Function syntax

```coldfusion
FileClose(fileObj)
```

See also

FileCopy, FileIsEOF, FileOpen, FileRead, FileReadLine, FileWrite

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileobj</td>
<td>The file to close.</td>
</tr>
</tbody>
</table>

Usage

Always close a file after opening it. When you use the FileOpen function to open a file, the file stream is opened and contents are read from or written to it. The FileClose function closes the stream. If you do not close a file, the stream remains open; in that case, the operating system can lock the file, which results in the file not being usable until the server is restarted.

Example

The following example checks to see if a file is still open and closes it.
<cfscript>
myfile = FileOpen("c:\ColdFusion9\wwwroot\test1.txt", "read");
while(NOT FileIsEOF(myfile))
{
x = FileReadLine(myfile);
WriteOutput("#x# <br>");
}
</cfscript>
<!--- Additional code goes here. --->
<cfif #myfile.status# IS "open">
<cfoutput>The file #myfile.filepath# is #myfile.status#</cfoutput><br>
<cfscript>
FileClose(myfile);
</cfscript>
</cfif>
**FileCopy**

**Description**

Copies the specified on-disk or in-memory source file to the specified destination file.

**Category**

*System functions*

**Function syntax**

```
FileCopy(source, destination)
```

**See also**

`FileClose`, `FileIsEOF`, `FileOpen`, `FileRead`, `FileReadLine`, `FileWrite`, `cffile`

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Pathname of the on-disk or in-memory file to copy. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the <code>getTempDirectory</code> function.</td>
</tr>
<tr>
<td>destination</td>
<td>Pathname of an on-disk or in-memory directory or file on the web server where the file is copied. If you specify a filename without a directory path, ColdFusion copies it relative to the source directory.</td>
</tr>
</tbody>
</table>

**Usage**

Use the following syntax to specify an in-memory file or directory, which is not written to disk. In-memory files speed processing of transient data.

```
ram:///filepath
```

The filepath can include directories, for example `ram:///petStore/images/poodle.jpg`. Create the directories in the path before you specify the file. For more information on using in-memory files, see [Working with in-memory files](#) in the *Developing ColdFusion Applications*.

**Example**
The following example copies the test1.txt file from the c:\testingdir\ directory to the c:\productiondir\ directory in Windows and names the new copy of the file test2.txt:

```
<h3>FileCopy Example</h3>
<cfset sourcefile="c:\testingdir\test1.txt">
<cfset destinationfile="c:\productiondir\test2.txt">

<cfif FileExists(#sourcefile#)>
    <cfif FileExists(#destinationfile#)>
        <cfoutput>A copy of #destinationfile# already exists.</cfoutput>
    <cfelse>
        <cfscript>
            FileCopy(#sourcefile#, #destinationfile#);
        </cfscript>
        <cfoutput>Copied: #sourcefile#<br>To: #destinationfile#</cfoutput>
    </cfif>
<cfelse>
    <cfoutput>The source file does not exist.</cfoutput><br>
</cfif>
```
**FileDelete**

**Description**

Deletes the specified on-disk or in-memory file on the server.

**Category**

*System functions*

**Function syntax**

```
FileDelete(filepath)
```

**See also**

`FileClose`, `FileIsEOF`, `FileOpen`, `FileRead`, `FileReadLine`, `FileWrite`, `cffile`

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filepath</td>
<td>Pathname of the on-disk or in-memory file to delete. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the <code>getTempDirectory</code> function.</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to free the memory used by an in-memory file. For more information on using in-memory files, see **Working with in-memory files** in the *Developing ColdFusion Applications*.

**Example**

The following example deletes the file `c:\productiondir\test1.txt` before moving `c:\testdir\test1.txt`:
<h3>FileDelete Example</h3>

<cfset sourcefile="c:\testdir\test1.txt">
<cfset destinationfile="c:\productiondir\test1.txt">

<cfif FileExists(#sourcefile#)>
  <cfif FileExists(#destinationfile#)>
    <cfoutput>The destination file already exists.<br>
    Deleting previous copy of #destinationfile#. <br>
    Moving: #sourcefile#<br>
    To: <br> #destinationfile#. </cfoutput><br>
  </cfoutput>
  <cfscript>
    FileDelete(#destinationfile#);
    FileMove(#sourcefile#, #destinationfile#);
  </cfscript>
</cfif>
<cfelse>
  <cfscript>
    FileMove(#sourcefile#, #destinationfile#);
  </cfscript>
</cfif>
<cfoutput>Moved: #sourcefile#<br>
To: <br> #destinationfile#. </cfoutput><br>
</cfif>
<cfelse>
  <cfoutput>The source file does not exist. </cfoutput><br>
</cfif>
FileExists

Description

Determines whether an on-disk or in-memory file exists.

Returns

Yes, if the file specified in the parameter exists; No, otherwise.

Category

System functions, Decision functions

Function syntax

FileExists(absolute_path)

See also

DirectoryExists, ExpandPath, GetTemplatePath

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolute_path</td>
<td>The absolute path of the on-disk or in-memory file.</td>
</tr>
</tbody>
</table>

Usage

To access a file on a remote system, the account (for Windows) or user (for UNIX and Linux) that is running ColdFusion must have permission to access the file, directory, and remote system. For example, if you run ColdFusion in the Server Configuration as a Windows service, by default it runs under the local system account, which does not have sufficient privileges to access remote systems. You can change this, however, on the Log On page of the Services Properties dialog box.

Example
<h3>FileExists Example</h3>

<cfset thisPath = ExpandPath("*.*")>
<cfset thisDirectory = GetDirectoryFromPath(thisPath)>
<cfoutput>
The current directory is: #GetDirectoryFromPath(thisPath)#
<cfif IsDefined("FORM.yourFile")>
<cfif FORM.yourFile is not ">
<cfset yourFile = FORM.yourFile>
<cfif FileExists(ExpandPath(yourfile))>
<p>Your file exists in this directory. You entered
the correct filename, #GetFileFromPath("#thisPath#/yourfile")#</p>
<cfelse>
FileGetMimeType

Description

Gets the MIME type for the file path/file object you have specified.

Returns

Returns MIME type.

Syntax

```
{{fileGetMimeType(path, strict)
fileGetMimeType(fileObject,strict)}}
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Full path on disk to the file if strict is set to true. If you do not specify the full path, the file is assumed to be present in the temp directory, as returned by the function <code>getTempDirectory</code>.</td>
</tr>
<tr>
<td>fileObject</td>
<td>Name of the file object.</td>
</tr>
<tr>
<td>strict</td>
<td>If false, determines the file type by extension. The default value is true.</td>
</tr>
</tbody>
</table>

Example

Assume that you have a file named test.pdf in /folder1 and test.txt in the same folder, and you want to check the MIME type. Here test.txt is a copy of test.pdf with extension renamed to txt.

```cfscript
<cfscript>
//Case 1.
mime.mimeType1 = FileGetMimeType(expandPath('/folder1/test.pdf'));
//Case 2.
mime.mimeType2 = FileGetMimeType(expandPath('/folder1/test.pdf'),false);
//Case 3.
mime.mimeType3 = FileGetMimeType(expandPath('/folder1/test.txt'));
//Case 4.
mime.mimeType4 = FileGetMimeType(expandPath('/folder1/test.txt'),false);
writeDump(mime);
</cfscript>
```

- **Case 1 and Case 2**: Returns application/pdf no matter if strict = true or false because the file is originally a PDF file.
- **Case 3**: Returns application/pdf since by default{ strict = true} and the file is originally a PDF which is renamed as TXT.
- **Case 4**: Returns text/plain since strict = false.
FileIsEOF

Description

Determines whether ColdFusion has reached the end of an on-disk or in-memory file while reading it.

Returns

Yes, if the end of the file has been reached; No, otherwise.

Category

System functions, Decision functions

Function syntax

FileIsEOF(fileObj)

See also

FileClose, FileOpen, FileRead, FileReadLine

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileobj</td>
<td>The file object.</td>
</tr>
</tbody>
</table>

Example

The following example reads a file until it reaches the end of the file:

```cfs
<h3>FileIsEOF Example</h3>

<cfscript>
myfile = FileOpen("c:\ColdFusion9\wwwroot\test1.txt", "read");
while(NOT FileIsEOF(myfile))
{
  x = FileReadLine(myfile);
  WriteOutput("#x# <br>");
}
FileClose(myfile);
</cfscript>
```
FileMove

Description

Moves an on-disk or in-memory file from one location to another on the server.

Category

System functions

History

ColdFusion 8: Added this function.

Function syntax

```
FileMove(source, destination)
```

See also

FileClose, FileCopy, FileOpen, FileRead, FileReadLine, FileWrite, cffile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Pathname of the on-disk or in-memory file to move. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the getTempDirectory function.</td>
</tr>
<tr>
<td>destination</td>
<td>Pathname of the destination on-disk or in-memory directory or file. If not an absolute path, it is relative to the source directory.</td>
</tr>
</tbody>
</table>

Usage

Use the following syntax to specify an in-memory file, which is not written to disk. In-memory files speed processing of transient data.

```
ram://filepath
```

The filepath can include directories, for example `ram://petStore/images/poodle.jpg`. Create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory
files in the Developing ColdFusion Applications.

Example

The following example moves the test1.txt file from the c:\testingdir\ directory to the c:\productiondir\ directory in Windows and renames the file test2.txt:

```
<h3>FileMove Example</h3>
<cfset sourcefile="c:\testingdir\test1.txt">
<cfset destinationfile="c:\productiondir\test2.txt">

<cfif FileExists(#sourcefile#)>
  <cfif FileExists(#destinationfile#)>
    <cfoutput>The destination file already exists.</cfoutput>
  </cfif>
  <cfelse>
    <cfscript>
      FileMove(#sourcefile#, #destinationfile#);
    </cfscript>
    <cfoutput>Moved: #sourcefile# <br>To: #destinationfile#.</cfoutput>
  </cfif>
</cfelse>

<cfelse>
  <cfoutput>The source file does not exist.</cfoutput>
</cfif>
```
FileOpen

Description

Opens an on-disk or in-memory file to read, write, or append. Use this function with the `FileRead` function to read large files.

Returns

A file object that represents the open file.

Category

System functions

Function syntax

```
FileOpen(filepath, [mode, charset])
```

See also

`FileClose`, `FileCopy`, `FileReadBinary`, `FileRead`, `FileReadLine`, `FileWrite`, `cffile`

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filepath</td>
<td>An absolute path of an on-disk or in-memory file on the server.</td>
</tr>
</tbody>
</table>
| mode      | Action to perform on the file, including the following:  
|           | • read  
|           | • readBinary  
|           | • write  
|           | • append  
|           | If you do not specify the mode, ColdFusion opens the file in read mode. |
| charset   | The character set of the file. |

Usage

The file does not have to exist before you open it. To write a new file, open it for writing, and then write it. The file object is a handle to a file. You can use the object as a structure to access the following information:

- filenameName of the file you opened
- filepathAbsolute path and filename
lastmodified: The time when the file was most recently modified
mode: The action for which the file was opened
size: The file size in bytes
status: Whether the file object is open or closed

The following opens a file, and then displays the absolute path and filename of that file:

```cfscript
myfile = FileOpen("c:\temp\test1.txt", "read");
</cfscript>
myfile refers to:
<cfdump var="#myfile.filepath#"

Use the following syntax to specify an in-memory file, which is not written to disk. In-memory files speed processing of transient data.

```ram:///filepath```

The filepath can include directories, for example `ram:///petStore/images/poodle.jpg`. Create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in the Developing ColdFusion Applications. Always close a file after opening it. When you use the `FileOpen` function to open a file, the file stream from the disk is opened and contents are read from or written to it. The `FileClose` function closes the stream. If you do not close a file, the stream remains open; in that case, the operating system can lock the file, which results in the file not being usable until the server is restarted.

Example

The following example opens a file, reads and outputs each line of the file, then closes the file.

```<h3>FileOpen Example</h3>
<cfscript>
myfile = FileOpen("c:\temp\test1.txt", "read");
while(NOT FileIsEOF(myfile))
{
x = FileReadLine(myfile);
WriteOutput("#x# <br>"); }
FileClose(myfile);
</cfscript>```
**FileRead**

**Description**

Reads an on-disk or in-memory text file or a file object created with the `FileOpen` function. You use this function either as an alternative to the `cf.file` tag with the `action="read"` attribute. You can also use this function to improve performance when reading a large file, because `FileRead` does not read the entire file into memory.

**Returns**

If you specify a filepath, the full text content of the file. If you specify a file object, the character or byte buffer of the specified size. If the file was opened in read mode, `FileRead` returns the character data (a string), otherwise it returns binary data.

**Category**

*System functions*

**Function syntax**

```
FileRead(filepath [, charset])
OR
FileRead(fileobj [, buffersize])
```

**See also**

`FileClose`, `FileIsEOF`, `FileReadBinary`, `FileReadLine`, `FileWrite`

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filepath</td>
<td>An absolute path to an on-disk or in-memory text file on the server.</td>
</tr>
</tbody>
</table>
charset

The character encoding in which the file contents is encoded. The following list includes commonly used values:

- utf-8
- iso-8859-1
- windows-1252
- us-ascii
- shift_jis
- iso-2022- jp
- euc-jp
- euc-kr
- big5
- euc-cn
- utf-16

If the file starts with a byte order mark and you set this attribute to a conflicting character encoding, ColdFusion generates an error.

fileobj

The file object from which to read.

buffersize

The number of characters to read.

Usage

You can read a text file or a file object with the FileRead function. When you specify an absolute path of a text file, ColdFusion reads the entire contents of the file. When you specify a file object, which you created using the FileOpen function, ColdFusion reads the number of characters specified in buffersize.

Example

```cfc
<h3>FileRead Example - Reading a file</h3>

<!--- This reads and outputs the entire file contents. --->
<cfscript>
myfile = FileRead("c:\temp\myfile.txt");
WriteOutput("#myfile#");
</cfscript>

<!--- This reads and outputs the first 100 characters --->
<!--- from the same file. --->
<cfscript>
myfile = FileOpen("c:\temp\test1.txt", "read");
x = FileRead(myfile, 100);
WriteOutput("#x#");
FileClose(myfile);
</cfscript>
```
FileReadBinary

Description

Reads an on-disk or in-memory binary file (such as an executable or image file) on the server, into a binary object parameter that you can use in the page. To send it through a web protocol (such as HTTP or SMTP) or store it in a database, first convert it to Base64 by using the `ToBase64` function.

⚠️ Note

This action reads the file into a variable in the local Variables scope. It is not intended for use with large files, such as logs, because they can bring down the server.

Returns

The entire contents of a binary file.

Category

System functions

Function syntax

```
FileReadBinary(filepath)
```

See also

`FileClose`, `FileIsEOF`, `FileRead`, `FileReadLine`, `FileWrite`

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filepath</td>
<td>An absolute path to an on-disk or in-memory binary file on the server</td>
</tr>
</tbody>
</table>

Usage

You convert the binary file to Base64 to transfer it to another site. ColdFusion 8 supports reading an image file as a binary and passing the result to a `cfimage`.

Example

The following example reads a binary file.
<h3>FileReadBinary Example</h3>
<cfscript>
myfile = FileReadBinary("c:\testingdir\test3.jpg");
</cfscript>
FileReadLine

Description

Reads a line from an on-disk or in-memory file.

Returns

The line of the file.

Category

System functions

Function syntax

```
FileReadLine(fileObj)
```

See also

FileClose, FileIsEOF, FileRead, FileWrite

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileobj</td>
<td>The file object</td>
</tr>
</tbody>
</table>

Example

The following example opens a file, reads each line, outputs each line, and then closes the file.

```
<h3>FileReadLIne Example</h3>
<cfscript>
myfile = FileOpen("c:\ColdFusion9\wwwroot\test1.txt", "read");
while(NOT FileIsEOF(myfile))
{
    x = FileReadLine(myfile); // read line
    WriteOutput("#x#");
}
FileClose(myfile);
</cfscript>
```
FileSeek

Description

Seeks the position for read or write operation of an on-disk or in-memory file on the server.

Returns

Returns the file position within a stream where the next file operation occurs.

Category

System functions

Function syntax

FileSeek(fileObj, pos)

See also

FileCopy, FileDelete, FileExists, FileMove, FileSetAccessMode, FileSetAttribute, FileSkipBytes

History

ColdFusion 9: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileObj</td>
<td>The file object.</td>
</tr>
<tr>
<td>pos</td>
<td>The position in a file within a stream where the following read and write operation must occur.</td>
</tr>
</tbody>
</table>

Example
<cfscript>
NewFile = FileOpen(ExpandPath(".") & "\test.txt","write","",true);
FileSeek(#NewFile#,0);
FileWrite(#NewFile#,"Hello World.. This is for FileOpen, FileSeek, FileSkipBytes.");
FileClose(#NewFile#);
WriteOutput("<br>Opening in Read Mode.<br>");
NewFile = FileOpen(ExpandPath(".") & "\test.txt","read","",true);
ReadFile = FileRead(#NewFile#,100);
WriteOutput("#ReadFile#<br>");
FileClose(#NewFile#);
WriteOutput("<br>Opening in Read-Write Mode.<br>");
NewFile = FileOpen(ExpandPath(".") & "\test.txt","readwrite","",true);
FileSeek(#NewFile#,2);
FileSkipBytes(#NewFile#,4);
FileWrite(#NewFile#,"Earth");
ReadFile = FileRead(#NewFile#,100);
WriteOutput("#ToString(ReadFile)#<br>");
FileClose(#NewFile#);
</cfscript>
FileSetAccessMode

Description

Sets the attributes of an on-disk file on UNIX or Linux. This function does not work with in-memory files.

Category

**System functions**

Function syntax

```
FileSetAccessMode(filepath, mode)
```

See also

**FileCopy, FileDelete, FileExists, FileMove**

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filepath</td>
<td>An absolute path to the file on the server.</td>
</tr>
<tr>
<td>mode</td>
<td>A three-digit value, in which each digit specifies the file access for individuals and groups:</td>
</tr>
</tbody>
</table>

  - The first digit represents the owner.
  - The second digit represents a group.
  - The third digit represents anyone.

  Each digit of this code sets permissions for the appropriate individual or group:

  - 4 specifies read permission.
  - 2 specifies write permission.
  - 1 specifies execute permission.

  You use the sums of these numbers to indicate combinations of the permissions:

  - 3 specifies write and execute permission.
  - 5 specifies read and execute permission.
  - 6 indicates read and write permission.
  - 7 indicates read, write, and execute permission.

  For example, 400 specifies that only the owner can read the file; 004 specifies that anyone can read the file.

Example
The following example sets the access mode of a file so that only the owner can read the file.

```<h3>FileSetAccessMode Example</h3>
<cfscript>
    FileSetAccessMode("test1.txt", "004");
</cfscript>```
**FileSetAttribute**

**Description**

Sets the attributes of an on-disk file in Windows. This function does not work with in-memory files.

**Category**

*System functions*

**Function syntax**

```
FileSetAttribute(filepath, attribute)
```

**See also**

*FileCopy, FileDelete, FileExists, FileMove*

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filepath</td>
<td>An absolute path to a file on the server.</td>
</tr>
</tbody>
</table>
| attribute | One of the following:  
  - `readOnly`  
  - `hidden`  
  - `normal`  
    Set the attribute to `normal` to make a file not read-only and not hidden. |

**Example**

The following example sets the access mode of a file to be read-only.

```
<h3>FileSetAttribute Example</h3>

<cfscript>
    FileSetAttribute("c:\temp\test1.txt", "readOnly");
</cfscript>
```
**FileSetLastModified**

**Description**

Sets the date when an on-disk or in-memory file was most recently modified.

**Category**

*System functions*

**Function syntax**

```
FileSetLastModified(filepath, date)
```

**See also**

`FileCopy`, `FileDelete`, `FileExists`, `FileMove`, `FileSetAccessMode`, `FileSetAttribute`

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filepath</td>
<td>An absolute path to an on-disk or in-memory file on the server.</td>
</tr>
<tr>
<td>date</td>
<td>A valid ColdFusion date or datetime.</td>
</tr>
</tbody>
</table>

**Example**

```
<cfscript>
    FileSetLastModified("c:\temp\test1.txt");
    WriteOutput(#GetFileInfo("c:\temp\test1.txt").lastmodified#);
</cfscript>
```
FileSkipBytes

Description

Skips over the data before a read or write operation of an on-disk or in-memory file on the server.

Category

System functions

Function syntax

FileSkipBytes(fileObj, noOfBytesToSkip)

See Also

FileCopy, FileDelete, FileExists, FileMove, FileSetAccessMode, FileSetAttribute, FileSeek

History

ColdFusion 9: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileobj</td>
<td>The file object.</td>
</tr>
<tr>
<td>noOfBytesToSkip</td>
<td>The number of bytes that must be skipped before the next file operation.</td>
</tr>
</tbody>
</table>

Usage

For noOfBytesToSkip, if you specify a value greater than the actual number of bytes, all bytes are skipped.

Example

See the Example section for the function FileSeek.
**FileUpload**

**Description**

Uploads file to a directory on the server.

**Returns**

Struct that contains the result (or status) of file upload. For details of what the struct contains, see the usage section of `cffile action = "upload`.

**Function syntax**

FileUpload(destination)
FileUpload(destination, filefield)
FileUpload(destination, filefield, accept)
FileUpload(destination, filefield, accept, nameconflict)

**History**

ColdFusion 9.0.1: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>Path of directory in which to upload the file. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory returned by the function <code>getTempDirectory</code>. If the destination you specify does not exist, ColdFusion creates a file with the specified destination name. For example, if you specify the destination C:\XYZ, ColdFusion creates a file XYZ in the C: drive.</td>
</tr>
<tr>
<td>accept</td>
<td>Limits the MIME types to accept. Comma-delimited list. For example, the following code permits JPEG and Microsoft Word file uploads: &quot;image/jpg,application/msword&quot;. The browser uses the filename extension to determine file type.</td>
</tr>
</tbody>
</table>
nameConflict

Action to take if file has the same name of a file in the directory.

- Error: File is not saved. ColdFusion stops processing the page and returns an error.
- Skip: File is not saved. This option permits custom behavior based on file properties.
- Overwrite: Replaces file.
- MakeUnique: Forms a unique filename for the upload; name is stored in the file object variable serverFile.

fileField

Name of form field used to select the file. Do not use number signs (#) to specify the field name.

See also

FileUploadAll

Usage

cfile action = "upload"

Example

```cfml
<cfif isdefined("form.fileData")>
  <cfscript>
    hello = FileUpload("<path>"","<mime type>"","unique");
  </cfscript>
  <cfdump var="#hello#">
<cfelse>
  <cfform name="myUpload" enctype="multipart/form-data">
    <cfinput type="file" name="fileData"><br>
    <cfinput type="submit" name="submit">
  </cfform>
</cfif>
```
**FileUploadAll**

**Description**

Uploads all files sent to the page in an HTTP request to a directory on the server.

**Returns**

An array of struct that provides the file upload status. For details of what the struct contains, see the Usage Section of `cffile action = "uploadAll"`.

**Function Syntax**

FileUploadAll(destination)
FileUploadAll(destination, accept)
FileUploadAll(destination, accept, nameConflict)

**History**

ColdFusion 11: Added ContinueOnError and Errors attribute.

ColdFusion 9.0.1: Added this function.

**Parameter**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>Path of directory in which to upload the file. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, returned by the function <code>getTempDirectory</code>. If the destination you specify does not exist, ColdFusion creates a file with the specified destination name. For example, if you specify the destination C:\XYZ, ColdFusion creates a file XYZ in the C: drive.</td>
</tr>
<tr>
<td>accept</td>
<td>Limits the MIME types to accept. Comma-delimited list. For example, the following code permits JPEG and Microsoft Word file uploads:({&quot;image/jpg, application/msword&quot;}) The browser uses the filename extension to determine file type.</td>
</tr>
</tbody>
</table>
### nameConflict

Action to take if file has the same name of a file in the directory.

- **Error**: File is not saved. ColdFusion stops processing the page and returns an error.
- **Skip**: File is not saved. Permits custom behavior based on file properties.
- **Overwrite**: Replaces file.
- **MakeUnique**: Forms a unique filename for the upload; name is stored in the file object variable `serverFile`.

### Added in ColdFusion 11:

<table>
<thead>
<tr>
<th>Value</th>
<th>Required</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| ContinueOnError| Optional | False   | By default, when uploading one of the files fail, the remaining files will not be uploaded. If this value is set to true, file upload continues even after encountering an upload error. A file upload error happens due to the following reasons:  
1. Empty file
2. Invalid file type
3. Invalid MIME or extension
4. File already exists  
In the case of an upload failure, the error details will be stored in the `errors` attribute. |
| Errors         | Optional | cfFile.uploadAllErrors | The name of the variable in which the file upload errors will be stored. Errors will be populated in the specified variable name when `continueOnError` is true. After the file upload is completed, this tag creates an array of structures that contains upload failure information for each upload failure. |
The upload failure information error structure contains the following fields:
• REASON - The reason for the failure
• DETAIL - File upload failure detail
• MESSAGE - A detailed message depicting the failure
• CLIENTFILE - Name of the file uploaded from the client's system
• CLIENTFILEEXT - Extension of the uploaded file on the client system (without a period)
• CLIENTFILENAME - Name of the uploaded file on the client system (without an extension)
• INVALID_FILE_TYPE - If the file mime type or extension is not in the specified accept attribute. If the reason is INVALID_FILE_TYPE, two additional keys will be available in the structure.
  • ACCEPT: list of mime types or file extensions given in the tag
  • MIMETYPE: mime type of the uploaded file
• EMPTY_FILE - If the uploaded file is an empty file
• FILE_EXISTS - If any file with the given name already exists in the destination and the overwritepolicy is error.
• DEST - The destination where file is copied
• FORM_FILE_NOT_FOUND - If the uploaded file is not found on the server

See also
FileUpload

Usage

cffile action = "uploadAll"
**FileWrite**

**Description**

If you specify a file path, writes the entire content to the specified on-disk or in-memory file. If you specify a file object, writes text or binary data to the file object.

**Category**

*System functions*

**Function syntax**

```
FileWrite(filepath, data [, charset])

OR

FileWrite(fileobj, data)
```

**See also**

*FileCopy, FileDelete, FileExists, FileMove, cffile*

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| charset   | The character encoding in which the file contents is encoded. The following list includes commonly used values:  
  - utf-8  
  - iso-8859-1  
  - windows-1252  
  - us-ascii  
  - shift_jis  
  - {{iso-2022-jp}}  
  - euc-jp  
  - euc-kr  
  - big5  
  - euc-cn  
  - utf-16  
  If the file starts with a byte order mark and you set this attribute to a conflicting character encoding, ColdFusion generates an error. |
| data      | Content of the file or file object to create. |
fileobj | Name of the file object to write.
filepath | Pathname of the on-disk or in-memory file to write. If not an absolute path (starting with a drive letter and a colon, or a forward or backward slash), it is relative to the ColdFusion temporary directory, which is returned by the getTempDirectory function.

Usage

Use the following syntax to specify an in-memory file, which is not written to disk. In-memory files speed processing of transient data.

```text
ram:///filepath
```

The filepath can include directories, for example `ram:///petStore/images/poodle.jpg`. Create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in the Developing ColdFusion Applications.

Example

```cfc
<h3>FileWrite Example</h3>
<!--- This example gets the email addresses of employees, --->
<!--- creates a file object that contains the e-mail addresses, --->
<!--- read the file object, and then creates a text file with a --->
<!--- list of e-mail addresses. --->

<cfquery name="getemployees" datasource="cfdocexamples">
SELECT EMAIL
FROM Employees
</cfquery>

<cfset companymail = ">
<cfloop query = "getemployees">
<cfset companymail = companymail & #EMAIL# & "; " & " ">
</cfloop>

<cfscript>
FileWrite("mail_list", 
#companymail#");
mlist = FileRead("mail_list");
FileWrite("c:\temp\mail_list.txt", 
#mlist#");
</cfscript>
```
**FileWriteLine**

**Description**

Appends the specified text to the file object.

**Category**

*System functions*

**Function syntax**

```plaintext
FileWriteLine(fileobj, text)
```

**See also**

`FileCopy, FileDelete, FileExists, FileMove, FileWrite, cffile`

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Content to add to the file object.</td>
</tr>
<tr>
<td>fileobj</td>
<td>The file object to which to write the line.</td>
</tr>
</tbody>
</table>

**Example**

```cfscript
<h3>FileWriteLine Example</h3>
<cfscript>
myfile = FileOpen("c:\temp\test1.txt", "write");
FileWriteLine(myfile,"This line is new.");
FileClose(myfile);
</cfscript>
```
Find

Description

Finds the first occurrence of a substring in a string, from a specified start position. The search is case sensitive.

Returns

A number; the position of substring in string; or 0, if substring is not in string.

Category

String functions

Function syntax

```
Find(substring, string [, start ])
```

See also

FindNoCase, Compare, FindOneOf, REFind, Replace

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>substring</td>
<td>A string or a variable that contains one. String for which to search.</td>
</tr>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String in which to search.</td>
</tr>
<tr>
<td>start</td>
<td>Start position of search.</td>
</tr>
</tbody>
</table>

Example

```
<coutput>
  <cfset stringToSearch = "The quick brown fox jumped over the lazy dog.">
  #find("the",stringToSearch)#<br>
  #find("the",stringToSearch,35)#<br>
  #find("no such substring",stringToSearch)#<br>
  <br>
  #findnocase("the",stringToSearch)#<br>
  #findnocase("the",stringToSearch,5)#<br>
  #findnocase("no such substring",stringToSearch)#<br>
  <br>
  #findoneof("aeiou",stringToSearch)#<br>
  #findoneof("aeiou",stringToSearch,4)#<br>
  #findoneof("%\^()",stringToSearch)#<br>
</cfoutput>
```
FindNoCase

Description

Finds the first occurrence of a substring in a string, from a specified start position. If substring is not in string, returns zero. The search is case-insensitive.

Returns

The position of substring in string; or 0, if substring is not in string.

Category

String functions

Function syntax

FindNoCase(substring, string [, start ])

See also

Find, CompareNoCase, FindOneOf, REFind, Replace

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>substring</td>
<td>A string or a variable that contains one. String for which to search.</td>
</tr>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String in which to search.</td>
</tr>
<tr>
<td>start</td>
<td>Start position of search.</td>
</tr>
</tbody>
</table>

Example

In the following example, the Find function returns 33 as the first position found because "the" is lowercase. The FindNoCase function returns 1 as the first position because the case is ignored.
<cfset stringToSearch = "The quick brown fox jumped over the lazy dog.">

stringToSearch = <cfoutput>#$stringToSearch#$</cfoutput><br>
<p>
Find Function:<br>
Find("the",stringToSearch) returns
<cfoutput>#$find("the",stringToSearch)#</cfoutput><br>
<p>
FindNoCase Function:<br>
FindNoCase("the",stringToSearch) returns
<cfoutput>#$FindNoCase("the",stringToSearch)#</cfoutput>
FindOneOf

Description

Finds the first occurrence of any one of a set of characters in a string, from a specified start position. The search is case sensitive.

Returns

The position of the first member of set found in string; or 0, if no member of set is found in string.

Category

String functions

Function syntax

```
FindOneOf(set, string [, start ])
```

See also

Find, Compare, REFind

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set</td>
<td>A string or a variable that contains one. String that contains one or more characters to search for.</td>
</tr>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String in which to search.</td>
</tr>
<tr>
<td>start</td>
<td>Start position of search.</td>
</tr>
</tbody>
</table>

Example
<cfoutput>
<cfset stringToSearch = "The quick brown fox jumped over the lazy dog.">
#find("the",stringToSearch)#<br>
#find("the",stringToSearch,35)#<br>
#find("no such substring",stringToSearch)#<br>
<br>
#findnocase("the",stringToSearch)#<br>
#findnocase("the",stringToSearch,5)#<br>
#findnocase("no such substring",stringToSearch)#<br>
<br>
#findoneof("aeiou",stringToSearch)#<br>
#findoneof("aeiou",stringToSearch,4)#<br>
#findoneof("@%^()",stringToSearch)#<br>
</cfoutput>
FirstDayOfMonth

Description
Determines the ordinal (day number, in the year) of the first day of the month in which a given date falls.

Returns
A number that corresponds to a day-number in a year.

Category
Date and time functions

Function syntax

FirstDayOfMonth(date)

See also
Day, DayOfWeek, DayOfWeekAsString, DayOfYear, DaysInMonth, DaysInYear

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD–9999 AD.</td>
</tr>
</tbody>
</table>

Usage
When passing a date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

Example

```cfoutput>
<h3>FirstDayOfMonth Example</h3>

The first day of #MonthAsString(Month(Now()))#, #Year(Now())# was day #FirstDayOfMonth(Now())# of the year.
</cfoutput>
```

© 2014 Adobe Systems Incorporated. All rights reserved.
Fix

Description

Converts a real number to an integer.

Returns

If `number` is greater than or equal to 0, the closest integer less than or equal to `number`. If `number` is less than 0, the closest integer greater than or equal to `number`.

Category

Mathematical functions

Function syntax

```
Fix(number)
```

See also

Ceiling, Int, Round

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A number</td>
</tr>
</tbody>
</table>

Example

```html
<h3>Fix Example</h3>
<p>Fix returns the closest integer less than the number if the number is greater than or equal to 0. Fix returns the closest integer greater than the number if number is less than 0.</p>
<cfoutput>
<p>The fix of 3.4 is #Fix(3.4)#</p>
<p>The fix of 3 is #Fix(3)#</p>
<p>The fix of 3.8 is #Fix(3.8)#</p>
<p>The fix of -4.2 is #Fix(-4.2)#</p>
</cfoutput>
```
FormatBaseN

Description

Converts *number* to a string, in the base specified by *radix*.

Returns

String that represents *number*, in the base *radix*.

Category

Display and formatting functions, Mathematical functions, String functions

Function syntax

FormatBaseN(number, radix)

See also

InputBaseN

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Number to convert</td>
</tr>
<tr>
<td>radix</td>
<td>Base of the result</td>
</tr>
</tbody>
</table>

Example
<h3>FormatBaseN Example</h3>
<p>Converts a number to a string in the base specified by Radix.</p>
<pre>
<cfoutput>
FormatBaseN(10, 2): #FormatBaseN(10, 2)#
FormatBaseN(1024, 16): #FormatBaseN(1024, 16)#
FormatBaseN(125, 10): #FormatBaseN(125, 10)#
FormatBaseN(10.75, 2): #FormatBaseN(10.75, 2)#
</cfoutput>
</pre>
<h3>InputBaseN Example</h3>
<p>InputBaseN returns the number obtained by converting a string, using base specified by Radix (an integer from 2 to 36).</p>
<pre>
<cfoutput>
InputBaseN("1010", 2): #InputBaseN("1010", 2)#
InputBaseN("3ff", 16): #InputBaseN("3ff", 16)#
InputBaseN("125", 10): #InputBaseN("125", 10)#
InputBaseN(1010, 2): #InputBaseN(1010, 2)#
</cfoutput>
</pre>
GenerateSecretKey

Description

Gets a secure key value for use in the Encrypt function.

Returns

A string that contains the encryption key.

Category

Security functions, String functions

Function syntax

```
GenerateSecretKey(algorithm [,keysize])
```

See also

Decrypt, Encrypt

History

ColdFusion 8: Added the keysiz attribute.
ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| algorithm | The encryption algorithm for which to generate the key. ColdFusion installs a cryptography library with the following algorithms:  
  - AES: the Advanced Encryption Standard specified by the National Institute of Standards and Technology (NIST) FIPS-197.  
  - BLOWFISH: the Blowfish algorithm defined by Bruce Schneier.  
  - DESEDE: the "Triple DES" algorithm defined by NIST FIPS-46-3. |
| keysize | Number of bits requested in the key for the specified algorithm. You can use this to request longer keys when allowed by the JDK. For example, the AES algorithm keys are limited to 128 bits unless the Java Unlimited Strength Jurisdiction Policy Files are installed. For more information, see http://java.sun.com/products/jce/index-14.html. |

**Usage**

You cannot use the `GenerateSecretKey` function to generate a key for the ColdFusion default encryption algorithm (CFMX_COMPAT) of the `Encrypt` and `Decrypt` functions.

ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, Adobe cannot provide technical support for third-party security providers.

**Example**

The following example encrypts and decrypts a text string. It lets you specify the encryption algorithm and encoding technique. It also has a field for a key seed to use with the CFMX_COMPAT algorithm. For all other algorithms, it uses the `GenerateSecretKey` function to generate a secret key.

```cfs
<h3>Decrypt Example</h3>

<!--- Do the following if the form has been submitted. --->
<cfif IsDefined("Form.myString")>
  <cfscript>
    /* GenerateSecretKey does not generate keys for the CFMX_COMPAT algorithm, so we use a key from the form. */
    if (Form.myAlgorithm EQ "CFMX_COMPAT")
      theKey=Form.MyKey;
    // For all other encryption techniques, generate a secret key.
    else
      theKey=generateSecretKey(Form.myAlgorithm);
    //Encrypt the string.
    encrypted=encrypt(Form.myString, theKey, Form.myAlgorithm, Form.myEncoding);
    //Decrypt it.
    decrypted=decrypt(encrypted, theKey, Form.myAlgorithm, Form.myEncoding);
  </cfscript>

  <!--- Display the values used for encryption and decryption, and the results. --->
  <cfoutput>
    <b>The algorithm:</b> #Form.myAlgorithm#<br>
    <b>The key:</b> #theKey#<br>
    <b>The string:</b> #Form.myString#<br>
    <b>Encrypted:</b> #encrypted#<br>
    <b>Decrypted:</b> #decrypted#<br>
  </cfoutput>
</cfif>
```
<!---- The input form. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
  <b>Select the encoding</b><br>
  <select size="1" name="myEncoding">
    <option selected>UU</option>
    <option>Base64</option>
    <option>Hex</option>
  </select><br>
  <br>
  <b>Select the algorithm</b><br>
  <select size="1" name="myAlgorithm">
    <option selected>CFMX_COMPAT</option>
    <option>AES</option>
    <option>DES</option>
    <option>DESEDE</option>
  </select><br>
  <br>
  <b>Input your key</b> (used for CFMX_COMPAT encryption only)<br>
  <input type = "Text" name = "myKey" value = "foobar"><br>
  <br>
  <b>Enter string to encrypt</b><br>
  <textarea name = "myString" cols = "40" rows = "5" WRAP = "VIRTUAL">This string will be encrypted (you can replace it with more typing).</textarea><br>
</form>
<input type="Submit" value="Encrypt my String" />
</form>
GetAuthUser

Description

Gets the name of an authenticated user.

Returns

The name of an authenticated user.

Category

Security functions

Function syntax

GetAuthUser()

See also

cflogin, cfloginuser, cflogout, GetUserRoles, IsUserInAnyRole, IsUserInRole, IsUserLoggedIn, ColdFusion security features in the Developing ColdFusion Applications

History

ColdFusion MX: Added this function.

Usage

This function works with cflogin authentication or web server authentication. It checks for a logged-in user as follows:

1. It checks for a login made with cfloginuser.
2. If no user was logged in with cfloginuser, it checks for a web server login (cgi.remote_user).

Example

<H3>GetAuthUser Example</H3>

<P>Authenticated User: <cfoutput>#GetAuthUser()#</cfoutput>
GetBaseTagData

Description

Used within a custom tag. Finds calling (ancestor) tag by name and accesses its data.

Returns

An object that contains data (variables, scopes, and so on) from an ancestor tag. If there is no ancestor by the specified name, or if the ancestor does not expose data (for example, cfif), an exception is thrown.

Category

Other functions

Function syntax

```
GetBaseTagData(tagname [, instancenumber ])
```

See also

GetBaseTagList; High-level data exchange in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagname</td>
<td>(Required) Ancestor tag name for which to return data</td>
</tr>
<tr>
<td>instancenumber</td>
<td>(Optional) Number of ancestor levels to jump before returning data. The default value is 1 (closest ancestor).</td>
</tr>
</tbody>
</table>

Example
<!--- This example shows the use of GetBaseTagData function. Typically used in custom tags.--->

...  
<cfif trim(inCustomTag) neq ">
  <cfoutput>
    Running in the context of a custom
tag named #inCustomTag#.<p>
  </cfoutput>
</cfif>
<!--- Get the tag instance data --->
<cfset tagData = GetBaseTagData(inCustomTag)>
<!--- Find the tag’s execution mode --->
Located inside the
<cfif tagData.thisTag.executionMode neq 'inactive'>
template
<cfelse>
BODY
</cfif>
GetBaseTagList

Description

Gets ancestor tag names, starting with the parent tag.

Returns

A comma-delimited list of uppercase ancestor tag names, as a string. The first list element is the current tag. If the current tag is nested, the next element is the parent tag. If the function is called for a top-level tag, it returns an empty string. If an ancestor does not expose data (see GetBaseTagData), its name might not be returned.

Category

Other functions

Function syntax

```
GetBaseTagList ()
```

See also

GetBaseTagData; High-level data exchange in the Developing ColdFusion Applications

Usage

This function does not display the following tags or end tags in the ancestor tag list:

- `cfif, cfelseif, cfelse`
- `cfswitch, cfcase, cfdefaultcase`
- `cftry, cfcatch`

This function displays the following tags only under the following conditions:

- `cfloop`: if it uses a query attribute
- `cfoutput`: if at least one of its children is a complex expression
- `cfprocessingdirective`: if it has at least one other attribute besides pageencoding

Example
<!--- This example shows the use of GetBaseTagList function. Typically used in custom tags. --->
<cfif thisTag.executionMode is "start">
<!--- Get the tag context stack
The list will look something like "CFIF,MYTAGNAME..." --->
<cfset ancestorList = GetBaseTagList()>
<br><br>Dump of GetBaseTagList output:<br>
<cfdump var="ancestorList"><br><br>
<!--- Output current tag name --->
<cfoutput>This is custom tag#ListGetAt(ancestorList,1)#</cfoutput><br>
<!--- Determine whether this is nested inside a loop --->
<cfset inLoop = ListFindNoCase(ancestorList, "cfloop")>
<cfif inLoop>
  Running in the context of a cfloop tag.<br>
</cfif>
</cfif>
</cfif>
GetBaseTemplatePath

Description

Gets the absolute path of an application’s base page.

Returns

The absolute path of the application base page, as a string.

Category

Other functions, System functions

Function syntax

GetBaseTemplatePath()

See also

GetCurrentTemplatePath, GetDirectoryFromPath, FileExists, ExpandPath

Usage

Contrast this with the GetCurrentTemplatePath function, which returns if called from an included page returns the page path of the included template rather than the base (including) template.

Example

```<h3>GetBaseTemplatePath Example</h3>
<p>The template path of the current page is: <cfoutput>#GetBaseTemplatePath()#</cfoutput></p>
<p>The directory containing the current template is: <cfoutput>#GetDirectoryFromPath(GetBaseTemplatePath())#</cfoutput></p>```
GetClientVariablesList

Description

Finds the client variables to which a page has write access.

Returns

Comma-delimited list of non-read-only client variables, as a string.

Category

List functions, Other functions

Function syntax

```
GetClientVariablesList()
```

See also

DeleteClientVariable

Usage

The list of variables returned by this function is compatible with ColdFusion list functions.

Example

```
<h3>GetClientVariablesList Example</h3>

<!--- The following line enables client variables. 
You would normally do this in Application.cfc.--->
<cfapplication clientmanagement="yes">
<p>This example creates two client variables and deletes the User_ID client variable,
if it exists in the list of client variables returned by GetClientVariablesList().</p>
<cfset client.somevar = ">
<cfset client.User_ID = ">
<p>Client variable list: <cfoutput>#GetClientVariablesList()#</cfoutput></p>
<cfif ListFindNoCase(GetClientVariablesList(), "User_ID") is not 0>
Delete client.User_ID variable.
    <cfset temp = DeleteClientVariable("User_ID")>
    <p>Was variable "User_ID" Deleted? <cfoutput>#temp#</cfoutput></p>
</cfif>
<p>Amended Client variable list: <cfoutput>#GetClientVariablesList()#</cfoutput>
```
GetComponentMetaData

Description

Gets metadata (such as the functions and implemented interfaces of a component) for a CFC or ColdFusion interface.

Returns

A structure containing the metadata for the CFC or interface. For information on the structure contents, see the component entry in the table in the GetMetaData Usage section.

Category

Extensibility functions

Function syntax

```plaintext
GetComponentMetaData(path)
```

See also

GetMetaData

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>The dot-delimited path of the interface or CFC definition. The path can be relative to the current directory or the web root. For example, if a cfm page that calls this function is in <code>web_root/my_apps/interface definitions</code>, the interface file is in <code>web_root/my_apps/interfaces/definitions</code>, and you want to get the metadata for the interface defined in <code>I2.cfc</code>, specify either of the following values in this parameter:</td>
</tr>
<tr>
<td></td>
<td>• <code>definitions.I2</code></td>
</tr>
<tr>
<td></td>
<td>• <code>my_apps.interface.definitions.I2.cfc</code></td>
</tr>
</tbody>
</table>

Usage

This function and the `getMetaData` function return the same data. This function, however, takes a path to the CFC or Interface definition file, and does not use or create an object instance. Also, this function can get data about CFCs and interfaces only, and you cannot specify an interface in the `getMetaData` function.
GetContextRoot

Description

Returns path to the JEE server context root for the current request.

Returns

The path from the web root to the context root for the current page. The path starts with a forward slash character but does not end with a forward slash character. For applications in the default (root) context, returns the empty string.

Category

System functions

History

ColdFusion MX 7: Added this function.

Function syntax

GetContextRoot()

See also

GetPageContext

Usage

This function is equivalent to calling GetPageContext().getRequest().getContextPath(). On JEE configurations, it returns the path from the Web root to the JEE context root of the ColdFusion JEE application. On server configurations, it returns the empty string, because the context root is at the web root. This function is useful in applications that might be installed at varying JEE context roots.

Example

The ColdFusion Administrator uses the following line to get the location of the administrator directory:

```<cfset request.thisURL = "#getContextRoot()#/CFIDE/administrator/">```

The Administrator uses the returned value in places where it uses a URL to access Administrator resources, such as images, as in the following line:

```<a href="index.cfm"><img src="#request.thisURL#images/back.gif" width="16" height="16" border="0" alt=""></a>```
GetCPUUsage

Description

Gets the CPU usage with default or custom snapshot interval. The default interval is 1000 milli-seconds.

Syntax

getCPUUsage()
getCPUUsage(long ms)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long ms</td>
<td>Time in milli-seconds. This is the time delay between two snapshots.</td>
</tr>
</tbody>
</table>
GetCurrentTemplatePath

Description

Gets the path of the page that calls this function.

Returns

The absolute path of the page that contains the call to this function, as a string.

Category

System functions

Function syntax

```
GetCurrentTemplatePath()
```

See also

GetCurrentTemplatePath, GetDirectoryFromPath, File Exists, ExpandPath

Usage

If the function call is made from a page included with a `<cfinclude>` tag, this function returns the page path of an included page. Contrast this with the `GetBaseTemplatePath` function, which returns the path of the top-level page, even if it is called from an included page.

Example

```
<!--- This example uses GetCurrentTemplatePath to show the template path of the current page --->
<h3>GetCurrentTemplatePath Example</h3>
<p>The template path of the current page is:<br> <cfoutput>#GetCurrentTemplatePath()#</cfoutput>
<p>The directory containing the current template is:<br> <cfoutput>#GetDirectoryFromPath(GetCurrentTemplatePath())#</cfoutput>
```
GetDirectoryFromPath

Description

Extracts a directory from an absolute on-disk or in-memory path.

Returns

Absolute path, without the filename. The last character is a forward or backward slash, depending on the operating system.

Category

System functions

Function syntax

\[
\text{GetDirectoryFromPath(path)}
\]

See also

ExpandPath, GetFileFromPath

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Absolute on-disk or in-memory path.</td>
</tr>
</tbody>
</table>

Usage

Example
<h3>GetDirectoryFromPath Example</h3>
<cfset thisPath = ExpandPath(".*")>
<cfset thisDirectory = GetDirectoryFromPath(thisPath)>
<cfoutput>
The current directory is: #GetDirectoryFromPath(thisPath)#
<cfif IsDefined("FORM.yourFile")>
  <cfif FORM.yourFile is not ">"
    <cfset yourFile = FORM.yourFile>
    <cfif FileExists(ExpandPath(yourFile))>
      <p>Your file exists in this directory. You entered the correct filename, #GetFileFromPath("#thisPath#/"yourfile")#</p>
    <cfelse>
      <p>Your file was not found in this directory: </p>
      <br>
      Here is a list of the other files in this directory:
      <!--- use cfdirectory show directory, order by name & size --->
      <cfdirectory directory = "#thisDirectory#"
        name = "myDirectory" SORT = "name ASC, size DESC">
      <!--- Output the contents of the cfdirectory as a CFTABLE --->
      <cftable query = "myDirectory">
        <cfcol header = "NAME:" text = "#Name#">
        <cfcol header = "SIZE:" text = "#Size#">
      </cftable>
    </cfif>
  </cfif>
</cfif>
</cfoutput>
<form action="getdirectoryfrompath.cfm" METHOD="post">
  <H3>Enter the name of a file in this directory <I><FONT SIZE="-1">(try expandpath.cfm)</FONT></I></H3>
  <input type="Text" NAME="yourFile">
  <input type="Submit" NAME="">
</form>
GetEncoding

Description

Returns the encoding (character set) of the Form or URL scope.

Returns

String; the character encoding of the specified scope.

Category

International functions, System functions

Function syntax

GetEncoding(scope_name)

See also

cfcontent, cfprocessingdirective, URLEncode, URLEncodedFormat

History

ColdFusion MX: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| scope_name  | • Form  
             • URL.             |

Usage

Use this function to determine the character encoding of the URL query string or the fields of a form that was submitted to the current page. The default encoding, if none has been explicitly set, is UTF-8. For more information, see [www.iana.org/assignments/character-sets](http://www.iana.org/assignments/character-sets).

Example
<!--- This example sends the contents of two fields and interprets them as big5 encoded text. Note that the form fields are received as URL variables because the form uses the GET method.--->
<cfcontent type="text/html; charset=big5">
<form action='#cgi.script_name#' method='get'>
<input name='xxx' type='text'>
<input name='yyy' type='text'>
<input type="Submit" value="Submit">
</form>

<cfif IsDefined("URL.xxx")>
<cfscript>
 SetEncoding("url", "big5");
 WriteOutput("URL.XXX is " & URL.xxx & "\n");
 WriteOutput("URL.YYY is " & URL.yyy & "\n");
 theEncoding = GetEncoding("URL");
 WriteOutput("The URL variables were decoded using '\" & theEncoding & '\" encoding.\n");

 WriteOutput("The encoding is " & theEncoding);
</cfscript>
</cfif>
**GetException**

**Description**


**Returns**

Any Java exception object raised by a previous method call on the Java object.

**Category**


**Syntax**

```
GetException(object)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A Java object.</td>
</tr>
</tbody>
</table>

**Usage**

ColdFusion stores a Java exception object for each method call on a Java object. Subsequent method calls reset the exception object. To get the current exception object, call `GetException` on the Java object before other methods are invoked on it.

**Example**

```
<!--- Create the Java object reference --->
<cfobject action = create type = java class = primativetype name = myObj>
<!--- Calls the object's constructor --->
<cfset void = myObj.init()>
<cftry>
    <cfset void = myObj.DoException() >
    <Cfcatch type = "Any">
        <cfset exception = GetException(myObj)>
    </Cfcatch>
    <!--- User can call any valid method on the exception object.--->
    <cfset message = exception.toString()>
    <cfoutput>
        Error<br>
        I got exception <br>
        <br> The exception message is: #message# <br>
    </cfoutput>
</cftry>
```

GetFileFromPath

Description

Extracts a filename from an absolute on-disk or in-memory path.

Returns

Filename, as a string.

Category

System functions

Function syntax

`GetFileFromPath(path)`

See also

`ExpandPath`, `GetCurrentTemplatePath`

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Absolute on-disk or in-memory path.</td>
</tr>
</tbody>
</table>

Example
<h3>GetFileFromPath Example</h3>
<cfset thisPath = ExpandPath("*.")>
<cfset thisDirectory = GetDirectoryFromPath(thisPath)>
<cfoutput>
The current directory is: #GetDirectoryFromPath(thisPath)#
<cfif IsDefined("FORM.yourFile")>
<cfif FORM.yourFile is not ">
<cfset yourFile = FORM.yourFile>
<cfif FileExists(ExpandPath(yourfile))>
<p>Your file exists in this directory. You entered the correct file name, #GetFileFromPath("#thisPath#/yourfile")#</p>
<cfelse>
<p>Your file was not found in this directory: #thisDirectory#</p>
<cfif cfdirectory.DIRECTORY = ":myDirectory"
<cfif cfdirectory.OrderBy = "name ASC, size DESC">
<cfif cfdirectory.QUERY = "myDirectory">
<cfif cfcollist.HEADER = "NAME:" TEXT = "#Name#">
<cfif cfcollist.HEADER = "SIZE:" TEXT = "#Size#">
...
GetFileInfo

Description

Retrieves information about on-disk or in-memory file.

Returns

The filename, path, parent directory, type, size, when the file was most recently modified, whether the file has read permission, write permission, and is hidden.

Category

System functions

Function syntax

```
GetFileInfo(path)
```

See also

FileOpen

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Absolute on-disk or in-memory path.</td>
</tr>
</tbody>
</table>

Usage

The function returns a structure that includes the following keys:

- Name: name of the file
- Path: absolute path of the file
- Parent: path to the file’s parent directory
- Type: either “directory” or “file”
- Size: file size in bytes
- Lastmodified: datetime when it was the file was most recently modified
- canRead: whether the file can be read
- canWrite: whether the file has write permission
- isHidden: whether the file is a hidden

Example
<cfscript>
    FileSetLastModified("c:\temp\test1.txt", "#Now('#'"));
    WriteOutput(GetFileInfo("c:\temp\test1.txt").lastmodified);
</cfscript>
GetFreeSpace

Description

Gets information about free hard disk space or free in-memory VFS space.

Returns

Returns the free space in bytes.

Syntax

getFreeSpace(path)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Path to the</td>
</tr>
<tr>
<td></td>
<td>• Hard disk drive, for example C: for Windows</td>
</tr>
<tr>
<td></td>
<td>or / for UNIX. Note that for Windows, you have</td>
</tr>
<tr>
<td></td>
<td>to explicitly use colon (:) and not just c. Only the</td>
</tr>
<tr>
<td></td>
<td>space of the entire hard disk drive is returned. If you</td>
</tr>
<tr>
<td></td>
<td>mention a directory in the path, for example, /opt/,</td>
</tr>
<tr>
<td></td>
<td>it is ignored and only / is considered.</td>
</tr>
<tr>
<td></td>
<td>• For in-memory file system, the path is ram:</td>
</tr>
</tbody>
</table>

Usage

See the usage section for GetTotalSpace.

Example

In the following example, in-memory file system memory for the application is set to 20 MB in ColdFusion Administrator. The function returns 20, which means the total space considered is 20 MB. This is because the value specified in the ColdFusion Administrator (Memory Limit per Application for In-Memory Virtual File System) is lesser than the value specified in the Application.cfc (20 MB).Application.cfc

```
<cfcomponent>
  <cfset this.name = "vfs_total_space">
  <cfset this.inmemoryfilesystem.size = 30>
</cfcomponent>
```

space.cfm
<cftry>
  <cfset totalRAMSpace = getTotalSpace("ram:")>
  <cfset freeRAMSpace = getFreeSpace("ram:")>
  <cfset totalDiskSpace = getTotalSpace("c:")>
  <cfset freeDiskSpace = getFreeSpace("c:")>
  <cfoutput>
    Total Hard Disk Space = #DecimalFormat(totalDiskSpace / (1024 * 1024 * 1024))# GB
    Free Hard Disk Space = #DecimalFormat(freeDiskSpace / (1024 * 1024 * 1024))# GB
    Total Application RAM Memory = #DecimalFormat(totalRAMSpace / (1024 * 1024))# MB
    Free Application RAM Memory = #DecimalFormat(freeRAMSpace / (1024 * 1024))# MB
  </cfoutput>
  <cfcatch type="any">
    <cfoutput>
      #cfcatch.message#
      #cfcatch.detail#
    </cfoutput>
  </cfcatch>
</cftry>
GetFunctionCalledName

Description

Returns the name of the variable used to call a defined function.

Returns

Name of the variable.

Category

System functions

Function syntax

```
GetFunctionCalledName()
```

History

ColdFusion 9: Added this function

Usage

This function can be used to return data from CFCs by simulating getters and setters. This applies only if the CFC does not use implicit getters and setters provided by ColdFusion 9.

Example

The following example shows how you can use this function to return data without defining explicit setters and getters:
//callednamedemo.cfc
component
{
    variables.x1 = 1;
    variables.y1 = 2;
    function init()
    {
        return this;
    }
    function get()
    {
        var name = getFunctionCalledName();
        return variables[mid(name,4,len(name)-3)];
    }
    function set(value)
    {
        var name = getFunctionCalledName();
        variables[mid(name,4,len(name)-3)] = value;
    }
    this.getX1 = get;
    this.getY1 = get;
    this.setX1 = set;
    this.setY1 = set;
}
<!--- calledname.cfm --->
<cfscript>
    function test()
    {
        return getFunctionCalledName();
    }
    WriteOutput(test() & "<br>"); // test
    a = test;
    WriteOutput(variables.a() & "<br>"); // a
    o = new callednamedemo();
    // shows *real* methods get(), SetX1() and getY1(), etc.
    writeDump(o);
    o.setX1(10);
    o.setY1(20);
    WriteOutput(o.getX1() & "<br>"); // 10
    WriteOutput(o.getY1() & "<br>"); // 20 </cfscript>
GetFunctionList

Description
Displays a list of the functions that are available in ColdFusion.

Returns
A structure of functions.

Category
System functions

Function syntax

```
GetFunctionList()
```

Example
```
<!--- This example shows the use of GetFunctionList. --->
<cfset fList = GetFunctionList()>
<cfoutput>StructCount(fList) # functions<br><br></cfoutput>
<cfloop COLLECTION = "fList" ITEM = "key">
<cfoutput>#key#<br></cfoutput>
</cfloop>
```
GetGatewayHelper

**Description**

Gets a Java GatewayHelper object that provides methods and properties for use with a ColdFusion event gateway.

**Returns**

A Java GatewayHelper object.

**Category**

Extensibility functions

**Function syntax**

```
GetGatewayHelper(gatewayID)
```

**See also**

SendGatewayMessage

**History**

ColdFusion MX 7: Added the function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayID</td>
<td>Identifier of the gateway that provides the GatewayHelper object. Must be the Gateway ID of one of the ColdFusion event gateway instances configured on the ColdFusion Administrator Event Gateways section’s Gateways page.</td>
</tr>
</tbody>
</table>

**Usage**

The ColdFusion GetGatewayHelper function returns a Java GatewayHelper object that provides event gateway-specific helper methods and properties. To use this function, the event gateway must provide access to a class that implements the GatewayHelper class. For example, an instant messaging event gateway might make buddy-list management functions available in a GatewayHelper object. An event gateway listener CFC can get the gatewayID value from the CFEvent structure of the incoming message. You access the GatewayHelper object’s methods and properties using standard ColdFusion Java object access techniques. For more information, see The role of the GatewayHelper object in the Developing ColdFusion Applications.

**Example**

If an event gateway’s helper class includes an addBuddy method that takes a single String parameter, you could use the following code to get the GatewayHelper object and add a buddy to the buddies list:
<h3>GetGatewayHelper Example</h3>
<cfscript>
  myHelper = getGatewayHelper(myGatewayID);
  status = myHelper.addBuddy("j smith");
</cfscript>
GetHttpRequestData

Description

Makes HTTP request headers and body available to CFML pages. Useful for capturing SOAP request data, which can be delivered in an HTTP header.

Returns

A ColdFusion structure.

Category

System functions

Function syntax

GetHttpRequestData()

The function returns a structure that contains the following entries:

<table>
<thead>
<tr>
<th>Structure element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>Raw content from form submitted by client, in string or binary format. For content to be considered string data, the FORM request header &quot;CONTENT_TYPE&quot; must start with &quot;text/&quot; or be special case &quot;application/x-www-form-urlencoded&quot;. Other types are stored as a binary object.</td>
</tr>
<tr>
<td>headers</td>
<td>Structure that contains HTTP request headers as value pairs. Includes custom headers, such as SOAP requests.</td>
</tr>
<tr>
<td>method</td>
<td>String that contains the CGI variable Request_Method.</td>
</tr>
<tr>
<td>protocol</td>
<td>String that contains the Server_Protocol CGI variable.</td>
</tr>
</tbody>
</table>

Usage

To determine whether data is binary, use IsBinary(x.content). To convert data to a string value, if it can be displayed as a string, use ToString(x.content).

Example

The following example shows how this function can return HTTP header information.
<cfset x = GetHttpRequestData()>
<cfoutput>
<table cellpadding = "2" cellspacing = "2">
<tr>
    <td><b>HTTP Request item</b></td>
    <td><b>Value</b></td> </tr>
    <cfloop collection = #x.headers# item = "http_item">
        <tr>
            <td>#http_item#</td>
            <td>#StructFind(x.headers, http_item)#</td> </tr>
    </cfloop>
<tr>
    <td>request_method</td>
    <td>#x.method#</td></tr>
<tr>
    <td>server_protocol</td>
    <td>#x.protocol#</td></tr>
</table>
<b>http_content --- #x.content#</b>
</cfoutput>
GetHttpTimeString

Description

Gets the current time, in the Universal Time code (UTC).

Returns

The time, as a string, according to the HTTP standard described in RFC 1123 and its underlying RFC, 822. This format is commonly used in Internet protocols, including HTTP.

Category

Date and time functions, International functions

Function syntax

GetHttpTimeString(date_time_object)

See also

GetLocale, GetTimeZoneInfo, SetLocale

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date_time_object</td>
<td>A ColdFusion date-time object string or Java Date object</td>
</tr>
</tbody>
</table>

Usage

The time in the returned string is UTC, consistent with the HTTP standard.

Example

```cfc<br>
<cfoutput>
  #GetHttpTimeString(now())#
</cfoutput>
```
GetK2ServerDocCount

Description

This function is deprecated.

Returns

The number of collection metadata items stored in Verity collections.

Category

Full-text search functions, Query functions

Function syntax

```plaintext
GetK2ServerDocCount ()
```

See also

GetK2ServerDocCountLimit

History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases.
ColdFusion MX: Added this function.

Example

```plaintext
<cfoutput>GetK2ServerDocCount = $*#GetK2ServerDocCount () #*$</cfoutput>
```
GetK2ServerDocCountLimit

Description

This function is deprecated.

Returns

Number of collection metadata items that the K2 server permits, as an integer

Category

Full-text search functions, Query functions

Function syntax

GetK2ServerDocCountLimit()

History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases. ColdFusion MX: Added this function.

Usage

If a search generates a larger number of documents than the limit, ColdFusion puts a warning message in the Administrator and in the log file.

Example

GetLocale

Description

Gets the current ColdFusion geographic/language locale value. To set the default display format of date, time, number, and currency values in a ColdFusion application session, you use the SetLocale function.

Returns

The current locale value, as an English string. If a locale has a Java name and a name that ColdFusion used prior to the ColdFusion MX 7 release (for example, en_US and English (US)), ColdFusion returns the ColdFusion name (for example, English (US)).

Category

Display and formatting functions, International functions, System functions

Function syntax

GetLocale()

See also

GetLocaleDisplayName, SetLocale

History

ColdFusion MX 7: Added support for all Java locales and locale names. ColdFusion MX: Changed behavior to that described in usage.

Usage

This function returns the locale name as it is represented in ColdFusion; for example, Portuguese (Brazilian), or ca_ES. To get a locale name in the language of the locale, use the GetLocaleDisplayName function, which returns português (Brasil) and(Espanya). This function determines whether a locale value is set for ColdFusion. (The value is set with the SetLocale function.)

- If the ColdFusion locale value is present, the function returns it.
- If the ColdFusion locale has not been explicitly set, ColdFusion now determines whether the default locale of the ColdFusion server computer operating system is among the locale values it supports. (The default locale is stored in the user environment variables user.language and user.region.) If the default locale value is not supported, the function returns English (US). ColdFusion sets the locale in the JVM to this value; it persists until the server is restarted or it is reset with the SetLocale function.

This function does not access a web browser’s Accept-Language HTTP header setting.

⚠️ Note

When ColdFusion is started, it stores the supported locale values in the variable Server.ColdFusion.SupportedLocales. ColdFusion supports the locales supported by its Java runtime environment. The SupportedLocales value lists the Java names for the supported locales and the corresponding names that ColdFusion used prior to the ColdFusion MX 7 release.
Example

```html
<h3>Example: Using SetLocale and GetLocale</h3>
<cfoutput>
  <!--- For each new request, the locale gets reset to the JVM locale --->
  Initial locale's ColdFusion name: #GetLocale()#<br>
  <br>
  <!--- Do this only if the form was submitted. --->
  <cfif IsDefined("form.mylocale")>
    <b>Changing locale to #form.mylocale#</b><br>
    <br>
    <!--- Set the locale to the submitted value and save the old ColdFusion locale name. --->
    <cfset oldlocale=SetLocale("#form.mylocale#")>
    <!--- Get the current locale. It should have changed. --->
    New locale: #GetLocale()#<br>
  </cfif>
  <!--- Self-submitting form to select the new locale. --->
  <cfform>
    <h3>Please select the new locale:</h3>
    <cfselect name="mylocale">
      <!--- The server.coldfusion.supportedlocales variable is a list of all supported locale names. Use a list cfloop tag to create an HTML option tag for each name in the list. --->
      <cfloop index="i" list="#server.coldfusion.supportedlocales#">
        <option value="#i#">#i#</option>
      </cfloop>
    </cfselect><br>
    <br>
    <cfinput type="submit" name="submitit" value="Change Locale">
  </cfform>
</cfoutput>
```
GetLocaleDisplayName

Description

Gets a locale value and displays the name in a manner that is appropriate to a specific locale. By default, gets the current locale in the current locale's language.

Returns

The localized display name of the locale, in the language of the specified locale.

Category

Display and formatting functions, International functions, System functions

Function syntax

GetLocaleDisplayName([locale, inLocale])

See also

GetLocale, SetLocale

History

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>locale</td>
<td>The locale whose name you want. The default is the current ColdFusion locale, or if no ColdFusion locale has been set, the JVM locale.</td>
</tr>
<tr>
<td>inlocale</td>
<td>The locale in which to return the name. The default is the current ColdFusion locale, or if no ColdFusion locale has been set, the JVM locale.</td>
</tr>
</tbody>
</table>

Example

The following example expands on the GetLocale example to show the use of the GetLocaleDisplayName function to display locale names in the current locale and in other locales. It lets you select a locale from all supported locales, changes the ColdFusion locale to the selected locales, and displays the old and new locale names.
Example: Changing and Displaying Locales

<!-- For each new request, the locale gets reset to the JVM locale -->
Initial locale's ColdFusion name: #GetLocale()#<br>
Initial locale's display name: #GetLocaleDisplayName()#<br>
<br>
<!-- Do this only if the form was submitted. -->
<cfif IsDefined("form.mylocale")>
<b>Changing locale to #form.mylocale#</b><br>
<br>
<!--- Set the locale to the submitted value. SetLocale returns the old ColdFusion locale name. -->
<cfset oldlocale=SetLocale("#form.mylocale#")>
<!---- Get the current locale's ColdFusion name. It should have changed. -->
<cfset newlocale=GetLocale()>
New locale's ColdFusion name: #newlocale#<br>
New locale's display name in current locale: #GetLocaleDisplayName()#<br>
New locale's display name in old locale: #GetLocaleDisplayName(newlocale, oldlocale)#<br>
New locale's display name in en_US: #GetLocaleDisplayName(newlocale, "en_US")#<br>
<br>
Old locale's display name in current locale: #GetLocaleDisplayName(oldlocale)#<br>
Old locale's display name in en_US: #GetLocaleDisplayName(oldlocale, "en_US")#<br>
</cfif>

<!---- Self-submitting form to select the new locale. -->
<cfform>
<h3>Please select the new locale:</h3>
<cfselect name="mylocale">
<!-- The server.coldfusion.supportedlocales variable is a list of all supported locale names. Use a list cfloop tag to create an HTML option tag for each name in the list. -->
<cfloop index="i" list="#server.coldfusion.supportedlocales#">
<!-- In the select box, we use the US English display names for the locales. You can change en_US to your preferred locale. -->
<option value="#i#"+#GetLocaleDisplayName(i, "en_US")#</option>
</cfloop>
</cfselect><br>
<br>
<cfinput type="submit" name="submitit" value="Change Locale">
</cfform>
</cfoutput>

</body>
</html>
GetLocalHostIP

Description

Returns the localhost IP address, which is 127.0.0.1 for IPv4 and :1 for IPv6 addresses.

Returns

The localhost IP address.

Category

Decision functions

Function syntax

GetLocalHostIP()

See also

IsLocalHost

History

ColdFusion MX 7.01: Added this function.

Example

<h3>GetLocalHostIP Example</h3>

<cfoutput>The localhost IP address is #GetLocalHostIP()#.</cfoutput>
GetMetaData

Description

Gets metadata (such as the methods, properties, and parameters of a component) associated with an object that is deployed on the ColdFusion server.

Returns

Structured metadata information: for ColdFusion components (CFCs) and user-defined functions (UDFs), a structure; for query objects, an array of structures.

Category

System functions

Function syntax

```
GetMetaData(object)
```

See also

CreateObject, GetComponentMetaData, QueryAddColumn, QueryNew

History

ColdFusion MX 7: Added support for getting query object metadata.
ColdFusion MX: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A ColdFusion component, user-defined function, or query object. Within a CFC, the parameter can also specify the This scope.</td>
</tr>
</tbody>
</table>

Usage

This function provides information about application data, and lets applications dynamically determine the structure of an object and how to use it. This function is useful for CFCs and query objects. The metadata for a CFC includes information on the component and its functions, arguments, and properties. The getMetaData function also returns metadata for user-defined functions that are not part of CFCs. Use the GetComponentMetaData function to get information about ColdFusion interfaces, or about CFC definitions that you have not instantiated. The following table lists the data returned by the function for supported object types:

<table>
<thead>
<tr>
<th>Object</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Component</th>
<th>A structure containing the following fields:</th>
<th>Value of the <strong>cfcomponent</strong> tag displayname attribute, if any.</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayname</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extends</td>
<td>Metadata for the component’s ancestor component. Components that do not explicitly extend another component extend the WEB-INF.cftags.component.</td>
<td></td>
</tr>
<tr>
<td>fullname</td>
<td>The dot delimited path from the cf_webroot of the component.</td>
<td></td>
</tr>
<tr>
<td>functions</td>
<td>Array of metadata structures for the component’s functions.</td>
<td></td>
</tr>
<tr>
<td>hint</td>
<td>Value of the <strong>cfcomponent</strong> tag displayname attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>implements</td>
<td>A structure containing the metadata of the interfaces that the component implements. The key of the structure is the interface name, and the value is a structure containing the interface metadata.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Component name, including the period-delimited path from a component search root such as the web root or a directory specified in the administrator Custom Tag Paths page.</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>Value of the <strong>cfcomponent</strong> tag output attribute, if any</td>
<td></td>
</tr>
<tr>
<td>path</td>
<td>Absolute path to the component.</td>
<td></td>
</tr>
<tr>
<td>properties</td>
<td>Array of structures containing metadata specified by the component’s <strong>cfproperty</strong> tags, if any.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Always, component.</td>
<td></td>
</tr>
<tr>
<td>userMetadata</td>
<td>User-specified attributes in the <strong>cfc component</strong> tag</td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>A structure containing the following fields.</td>
<td>Value of the <code>cffunction</code> tag <code>access</code> attribute, if any.</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>displayname</td>
<td>Value of the <code>cffunction</code> tag <code>displayname</code> attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>hint</td>
<td>Value of the <code>cffunction</code> tag <code>hint</code> attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Function name.</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>Value of the <code>cffunction</code> tag <code>output</code> attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>parameters</td>
<td>Array of structures containing metadata for the function parameters.</td>
<td></td>
</tr>
<tr>
<td>returnformat</td>
<td>The format in which to return values to a remote caller. This attribute has no effect on values returned to a local caller.</td>
<td></td>
</tr>
<tr>
<td>returntype</td>
<td>Value of the <code>cffunction</code> tag <code>returntype</code> attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>roles</td>
<td>Value of the <code>cffunction</code> tag <code>output</code> attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>userMetadata</td>
<td>User-specified attributes in the <code>cffunction</code> tag.</td>
<td></td>
</tr>
<tr>
<td>Parameter or Property</td>
<td>A structure containing the following fields:</td>
<td>Value of the <code>cfargument</code> or <code>cfproperty</code> tag <code>default</code> attribute, if any.</td>
</tr>
<tr>
<td>displayname</td>
<td>Value of the <code>cfargument</code> or <code>cfproperty</code> tag <code>displayname</code> attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>hint</td>
<td>Value of the <code>cfargument</code> or <code>cfproperty</code> tag <code>hint</code> attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Function parameter or CFC property name.</td>
<td></td>
</tr>
<tr>
<td>required</td>
<td>Value of the <code>cfargument</code> or <code>cfproperty</code> tag required attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Value of the <code>cfargument</code> or <code>cfproperty</code> tag type attribute, if any.</td>
<td></td>
</tr>
<tr>
<td>userMetadata</td>
<td>User-specified attributes in the argument tag.</td>
<td></td>
</tr>
</tbody>
</table>
| Query | An array of structures containing the following elements:  
IsCaseSensitive  
  |
| name | The column name. |
| TypeName | The SQL data type (Omitted if the query object is created with QueryNew without specifying types.) |

⚠️ **Note**

Use the This scope to access component metadata inside the CFC. The This scope is available at runtime in the component body and in the CFC methods. It is used to read and write variables that are present during the life of the component.

For more information, see [Using introspection to get information about components](#) in the *Developing ColdFusion Applications*.

**Example**

The following example uses the `cfdump` tag to display metadata for the utilities CFC that is used by the ColdFusion component browser. It also displays the names and data types of the fields in the `cfdocexamples` database Employees table.
<cfscript>
    componentutils = createObject("component", "cfide.componentutils.utils");
    utilmetadata = getMetaData(componentutils);
</cfscript>

<h4>Metadata for the CFC component utilities</h4>
<cfdump var="#utilmetadata#">

<cfquery name="getemployees" datasource="cfdocexamples">
    SELECT *
    FROM Employees
</cfquery>
<cfset employeemeta=getMetaData(getemployees)>

<h4>The Employees table has the following columns</h4>
<cfloop index="i" from="1" to="#arrayLen(employeemeta)#">
    <cfoutput>
        #employeemeta[i].name# #employeemeta[i].TypeName# #employeemeta[i].isCaseSensitive#<br>
    </cfoutput>
</cfloop>
GetMetricData

Description

Gets server performance metrics.

Returns

ColdFusion structure that contains metric data, depending on the mode value.

Category

System functions

Function syntax

GetMetricData(mode)

History

ColdFusion MX: Deprecated the cachepops parameter. It might not work, and it might cause an error, in later releases.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>perf_monitor</td>
<td>Returns internal data, in a structure. To receive data, enable PerfMonitor in ColdFusion Administrator before executing the function. On Windows, this data is otherwise displayed in the Windows PerfMonitor.</td>
</tr>
<tr>
<td></td>
<td>simple_load</td>
<td>Returns an integer value that is computed from the state of the server’s internal queues. Indicates the overall server load.</td>
</tr>
<tr>
<td></td>
<td>prev_req_time</td>
<td>Returns the time, in milliseconds, that it took the server to process the previous request.</td>
</tr>
</tbody>
</table>
avg_req_time

Returns the average time, in milliseconds, that it takes the server to process a request. Changing the setting to 0 prevents the server from calculating the average and removes overhead associated with gathering data. The default value is 120 seconds.

Usage

If `mode = "perf_monitor"`, the function returns a structure with these data fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstanceName</td>
<td>The name of the ColdFusion server. The default value is cfserver.</td>
</tr>
<tr>
<td>PageHits</td>
<td>Number of HTTP requests received since ColdFusion was started.</td>
</tr>
<tr>
<td>ReqQueued</td>
<td>Number of HTTP requests in the staging queue, waiting for processing.</td>
</tr>
<tr>
<td>DBHits</td>
<td>Number of database requests since the server was started.</td>
</tr>
<tr>
<td>ReqRunning</td>
<td>Number of HTTP requests currently running. In the ColdFusion Administrator, you can set the maximum number of requests that run concurrently.</td>
</tr>
<tr>
<td>ReqTimedOut</td>
<td>Number of HTTP requests that timed out while in the staging queue or during processing.</td>
</tr>
<tr>
<td>BytesIn</td>
<td>Number of bytes in HTTP requests to ColdFusion.</td>
</tr>
<tr>
<td>BytesOut</td>
<td>Number of bytes in HTTP responses from ColdFusion.</td>
</tr>
<tr>
<td>AvgQueueTime</td>
<td>For the last two HTTP requests (current and previous), the average length of time the request waited in the staging queue.</td>
</tr>
<tr>
<td>AvgReqTime</td>
<td>For the last two HTTP requests (current and previous), the average length of time the server required to process the request.</td>
</tr>
<tr>
<td>AvgDBTime</td>
<td>For the last two HTTP requests (current and previous), the average length of time the server took to process CFQueries in the request.</td>
</tr>
</tbody>
</table>
Example

<!--- This example gets and displays metric data from Windows NT PerfMonitor --->
<cfset pmData = GetMetricData( "PERF_MONITOR" ) >
<cfoutput>
  Current PerfMonitor data is: <p>
  InstanceName: #pmData.InstanceName# <p>
  PageHits: #pmData.PageHits# <p>
  ReqQueued: #pmData.ReqQueued# <p>
  DBHits: #pmData.DBHits# <p>
  ReqRunning: #pmData.ReqRunning# <p>
  ReqTimedOut: #pmData.ReqTimedOut# <p>
  BytesIn: #pmData.BytesIn# <p>
  BytesOut: #pmData.BytesOut# <p>
  AvgQueueTime: #pmData.AvgQueueTime# <p>
  AvgReqTime: #pmData.AvgReqTime# <p>
  AvgDBTime: #pmData.AvgDBTime# <p>
</cfoutput>
GeneratePBKDFKey

*GeneratePBKDFKey* has been added in ColdFusion 11 to allow users to support PBKDF2 key derivation.

**Returns**

A string that contains the encryption key.

**History**

ColdFusion 11: Added this function

**Category**

Security functions, String functions

**Function syntax**

```
GeneratePBKDFKey(String algorithm, String string, String salt, int iterations, int keysize)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithm</td>
<td>The encryption algorithm for which to generate the key. The supported algorithms are PBKDF2WithHmacSHA1, PBKDF2WithSHA1, PBKDF2WithSHA224, PBKDF2WithSHA256, PBKDF2WithSHA384, PBKDF2WithSHA512</td>
</tr>
<tr>
<td>string</td>
<td>The string to be used for conversion</td>
</tr>
<tr>
<td>salt</td>
<td>A random salt. The standard recommends a salt length of at least 64 bits (8 characters). The salt needs to be generated using a pseudo random number generator (e.g SHA1PRNG).</td>
</tr>
<tr>
<td>iterations</td>
<td>The number of PBKDFE iterations to perform. The recommended value for iterations is 1000 or more.</td>
</tr>
<tr>
<td>keysize</td>
<td>The key size in number of bits.</td>
</tr>
</tbody>
</table>

**Example**

Encryption using PBKDF2:
<cfscript>
salt="A41n9t0Q"
password = "Password@123"
PBKDFalgorithm = "PBKDF2WithSHA224"
dataToEncrypt= "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua"
encryptionAlgorithm = "AES"
derivedKey = GeneratePBKDFKey(PBKDFalgorithm ,password ,salt,4096,128);
writeOutput("Generated PBKDFKey (Base 64) : " & derivedKey);
encryptedData = encrypt(dataToEncrypt, derivedKey, encryptionAlgorithm, "BASE64")
writeOutput("Data After Encryption using PBKDF2: " & encryptedData);
</cfscript>

Decryption using PBKDF2:

<cfscript>
salt="A41n9t0Q"
password = "Password@123"
PBKDFalgorithm = "PBKDF2WithSHA224"
derivedKey = GeneratePBKDFKey(PBKDFalgorithm ,password ,salt,4096,128)
decryptedData = decrypt(encryptedData, derivedKey, encryptionAlgorithm, "BASE64")
writeOutput("Data After Decryption using PBKDF2: " & decryptedData);
</cfscript>
GetPageContext

Description

Gets the current ColdFusion PageContext object that provides access to page attributes and configuration, request, and response objects.

Returns

The current ColdFusion Java PageContext Java object.

Category

System functions

Function syntax

GetPageContext()

History

ColdFusion MX: Added this function.

Usage

The ColdFusion PageContext class is a wrapper class for the Java PageContext object that can resolve scopes and perform case-insensitive variable lookups. The PageContext object exposes fields and methods that can be useful in J2EE integration. It includes the include and forward methods that provide the equivalent of the corresponding standard JSP tags. You use these methods to call JSP pages and servlets. For example, you use the following code in CFScript to include the JSP page hello.jsp and pass it a name parameter:

GetPageContext().include("hello.jsp?name=Bobby");

When you use GetPageContext to include a JSP page in a CFML page on WebLogic, you may need to flush the output of the CFML page with cfflush before calling the JSP page. Otherwise, the ColdFusion output appears after the JSP output. The methods supported on the returned PageContext are only the ones mandated by the JSP specification. To look up scopes by name, use the StructGet function, for example:

<cfset myscope = "server">
<cfset myserver = StructGet(myscope)>

For more information, see your Java Server Pages (JSP) documentation. On WebLogic, you may need to flush the output of the CFML page (using cfflush) before calling a JSP page. If you do not, the ColdFusion output appears after the JSP output. On JBOSS, if you use this function and you include any JSP file, the content of the JSP file is processed prior to the processing of any CFML tags before the function.

Example
<cfset pc = GetPageContext()>

<cfset pc.setAttribute("name","John Doe")>
<cfoutput>name: #variables.name#<br></cfoutput>

<cfoutput>Language of the current locale is
#pc.getRequest().getLocale().getDisplayLanguage()#</cfoutput>.
GetPrinterInfo

Description

Determines which print attributes are supported by a selected printer.

Returns

A structure that contains the attributes supported by the printer. If the printer is not specified, the structure contains attributes supported by the default printer, if one exists.

Category

System functions

Function syntax

```
GetPrinterInfo("printer")
```

See also

cfpdf, cfprint

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>printer</td>
<td>The name of a printer. An example in Windows is “\s1001prm02\NTN-2W-HP_BW02”. The default is the default printer for the account where the ColdFusion server is running. Printer names are case sensitive and must be entered exactly as they appear in the System Information page of the ColdFusion Administrator. If you specify an empty string, for example <code>GetPrinterInfo(&quot;&quot;)</code>, ColdFusion generates an error. Use the following code to retrieve information on the default printer:<code>{{ GetPrinterInfo().}}</code></td>
</tr>
</tbody>
</table>

Usage

Use this function in conjunction with the cfprint tag when processing large print jobs. Not all printers support the complete list of print attributes allowed by the cfprint tag. If the selected printer does not support a print attribute, the printer ignores the attribute. In Windows systems, the account running the ColdFusion server must have PRINTE
R_ACCESS_USE access rights for each printer it uses. Even if the printer is configured locally on the system, the printer is not available if the account in which ColdFusion is running does not have the proper permissions. For more information on attributeStruct, see cfprint.

Example

```coldfusion
<!--- The following code returns information on the default printer. --->
<cfdump var="#GetPrinterInfo()"/>

<!--- The following code returns information on the specified printer. --->
<cfdump var="#GetPrinterInfo('\\S1001prn02\NTN-2W-SHARP01')"/>
```
GetPrinterList

Description
Gets the list of printers that are accessible by ColdFusion server.

Returns
A list of accessible printers.

Category
System functions

Function Syntax

getPrinterList()

See Also
cfpdf, cfprint

History
ColdFusion 8: Added this function.

Usage
Ensure that you run ColdFusion server using an account that has appropriate permissions to access all desired printers.
GetProfileSections

Description

Gets all the sections of an initialization file. An initialization file assigns values to configuration variables, also known as entries, that are set when the system boots, the operating system comes up, or an application starts. An initialization file has the suffix INI; for example, boot.ini, Win32.ini.

Returns

An initialization file, as a structure whose format is as follows:

- Each initialization file section name is a key in the structure
- Each list of entries in a section of an initialization file is a value in the structure

If there is no value, returns an empty string.

History

ColdFusion 11: Added this function

Category

System functions

Function syntax

GetProfileSections(iniFile, [encoding])

See also

GetProfileString, SetProfileString

History

ColdFusion 11: Added the optional parameter, encoding.

ColdFusion MX: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iniFile</td>
<td>Absolute path (drive, directory, filename, extension) of initialization file; for example, C:\boot.ini</td>
</tr>
<tr>
<td>encoding</td>
<td>(Optional) Encoding of the initialization (ini) file. For instance &quot;UTF-8&quot; For a list of encodings, see cffile action = &quot;read&quot;.</td>
</tr>
</tbody>
</table>

Example
GetProfileSections(expandPath("./russian.ini"), "UTF-8");
GetProfileString

Description

Gets an initialization file entry. An initialization file assigns values to configuration variables, also known as entries, that are set when the system boots, the operating system comes up, or an application starts. An initialization file has the suffix INI; for example, boot.ini, Win32.ini.

Returns

An entry in an initialization file, as a string. If there is no value, returns an empty string.

History

ColdFusion 11: Added the encoding attribute.

Category

System functions

Function syntax

GetProfileString(iniPath, section, entry, [encoding])

See also

GetProfileSections, SetProfileString

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iniPath</td>
<td>Absolute path (drive, directory, filename, extension) of initialization file; for example, C:\boot.ini</td>
</tr>
<tr>
<td>section</td>
<td>Section of initialization file from which to extract information</td>
</tr>
<tr>
<td>entry</td>
<td>Name of value to get</td>
</tr>
<tr>
<td>encoding</td>
<td>Encoding of the initialization (ini) file. For instance &quot;UTF-8&quot;. For a list of encodings, see cffile action = &quot;read&quot;</td>
</tr>
</tbody>
</table>

Example
<h3>GetProfileString Example</h3>
Uses GetProfileString to get the value of timeout in an initialization file. Enter
the full path of your initialization file, and submit the form.
<!--- If the form was submitted, it gets the initialization path and timeout value
specified in the form --->
<cfif Isdefined("Form.Submit")>
<cfset IniPath = FORM.iniPath>
<cfset Section = "boot loader">
<cfset timeout = GetProfileString(IniPath, Section, "timeout")>
<h4>Boot Loader</h4>
<!--- If no entry in an initialization file, nothing displays --->
<p>Timeout is set to: <cfoutput>#timeout#</cfoutput>.</p>
</cfif>
<form action = "getprofilestring.cfm">
<table cellspacing = "2" cellpadding = "2" border = "0">
<tr>
<td>Full Path of Init File</td>
<td><input type = "Text" name = "IniPath" value = "C:\myboot.ini"></td>
</tr>
<tr>
<td><input type = "Submit" name = "Submit" value = "Submit"></td>
</tr>
</table>
</form>
GetReadableImageFormats

Description

Returns a list of image formats that ColdFusion can read on the operating system where ColdFusion is deployed.

Returns

A list of image file formats.

Category

System functions

History

ColdFusion 8: Added this function.

Function syntax

```
GetReadableImageFormats()
```

See also

GetWriteableImageFormats, cfimage (for supported image file formats)

Usage

Use this function to determine image file compatibility on the ColdFusion server.

Example

```
<cfoutput>#GetReadableImageFormats()#</cfoutput>
```
GetSOAPRequest

Description

Returns an XML object that contains the entire SOAP request. Usually called from within a web service CFC.

Returns

An XML object that contains the entire SOAP request.

Category

XML functions

History

ColdFusion MX 7: Added this function.

Function syntax

```coldfusion
GetSOAPRequest()
```

See also

AddSOAPRequestHeader, AddSOAPResponseHeader, GetSOAPRequestHeader, GetSOAPResponse, GetSOAPResponseHeader, IsSOAPRequest; Basic web service concepts in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>webservice</td>
<td>Optional. A webservice object as returned from the <code>cfoject</code> tag or the CreateObject function. Required if the function is called from the client.</td>
</tr>
</tbody>
</table>

Usage

Call this function to obtain a web service request object after the web service has been invoked. If you call this function from outside a web service CFC without the `webservice` parameter, it throws the following expression error:

```
Unable to use getSOAPRequest: not processing a web service request.
```

If you call this function from within a web service CFC, you can omit the `webservice` argument. The function executes against the request that it is currently processing. You can use CFML XML functions to examine the returned XML object.

Example

This example makes a request to execute the `echo_me` function of the headerservice.cfc web service. For
information on implementing the headerservice.cfc web service and also to see the `echo_me` function and the content of the web service CFC, see the example for either the `AddSOAPResponseHeader` function or the `GetSOAPRequestHeader` function.

```coldfusion
<!--- Note that you might need to modify the URL in the CreateObject function to match your server and the location of the headerservice.cfc file if it is different than shown here. Note, too, that getSOAPRequest is called from the client here, whereas often it would be called from within the web service CFC. --->

<cfscript>
  ws = CreateObject("webservice", "http://localhost/soapheaders/headerservice.cfc?WSDL");
  ws.echo_me("hello world");
  req = getSOAPRequest(ws);
</cfscript>
<cfdump var="#req#">
```
GetSOAPRequestHeader

Description

Obtains a SOAP request header. Call only from within a CFC web service function that is processing a request as a SOAP web service.

Returns

A SOAP request header.

Category

XML functions

History

ColdFusion MX 7: Added this function.

Function syntax

GetSOAPRequestHeader(namespace, name [, asXML])

See also

AddSOAPRequestHeader, AddSOAPResponseHeader, GetSOAPRequest, GetSOAPResponse, GetSOAPResponseHeader, IsSOAPRequest; Basic web service concepts in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>A String that is the namespace for the header.</td>
</tr>
<tr>
<td>name</td>
<td>A String that is the name of the header.</td>
</tr>
<tr>
<td>asXML</td>
<td>If true, the header is returned as a CFML XML object; if false (default), the header is returned as a Java object.</td>
</tr>
</tbody>
</table>

Usage

If you specify false for the asXML parameter, ColdFusion first attempts to retrieve the header using the data type specified in the header’s xsi:type attribute. If the xsi:type attribute is not available, ColdFusion attempts to retrieve the header as a string. If you specify true for the asXML parameter, ColdFusion retrieves the header as raw XML. This function throws an error if it is invoked in a context that is not a web service request. Use the IsSOAPRequest function to determine whether the CFC is running as a web service.

Example

This example creates a CFC web service that illustrates the operation of the GetSOAPRequestHeader function and also provides a web service that illustrates the operation of other ColdFusion SOAP functions. Save the
following code as headerservice.cfc in a folder called soapheaders under your web root. Test its operation, and specifically the operation of the `GetSOAPRequestHeader` function, by executing the examples that invoke this web service. For example, see the example for `AddSOAPRequestHeader`.

```cfc
cfcomponent displayName="tester" hint="Test for SOAP headers">

cffunction name="echo_me" access="remote" output="false" returntype="string" displayname="Echo Test" hint="Header test">
  <cfargument name="in_here" required="true" type="string">
  <cfset isSOAP = isSOAPRequest()>
  <cfif isSOAP>
    <!--- Get the first header as a string and as XML --->
    <cfset username = getSOAPRequestHeader("http://mynamespace/", "username")>
    <cfset return = "The service saw username: " & username>
    <cfset xmlusername = getSOAPRequestHeader("http://mynamespace/", "username", "TRUE")>
    <cfset return = return & "<br> as XML: " & xmlusername>
    <!--- Get the second header as a string and as XML --->
    <cfset password = getSOAPRequestHeader("http://mynamespace/", "password")>
    <cfset return = return & "The service saw password: " & password>
    <cfset xmlpassword = getSOAPRequestHeader("http://mynamespace/", "password", "TRUE")>
    <cfset return = return & "<br> as XML: " & xmlpassword>
    <!--- Add a header as a string --->
    <cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED VALUE", false)>
    <!--- Add a second header using a CFML XML value --->
    <cfset doc = XmlNew()>
    <cfset x = XmlElemNew(doc, "http://www.tomj.org/myns", "returnheader2")>
    <cfset x.XmlText = "hey man, here I am in XML">
    <cfsetx.XmlAttributes["xsi:type"] = "xsd:string">
    <cfset tmp = addSOAPResponseHeader("ignoredNameSpace", "ignoredName", x)>
    <cfelse>
    <!--- Add a header as a string - Must generate error! --->
    <cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED VALUE", false)>
  --->
  <cfset return = "Not invoked as a web service">
</cfif>
  <cfreturn return>
</cffunction>
</cfcomponent>
```
GetSOAPResponse

Description

Returns an XML object that contains the entire SOAP response after invoking a web service.

Returns

An XML object that contains the entire SOAP response.

Category

XML functions

History

ColdFusion MX 7: Added this function.

Function syntax

```
GetSOAPResponse(webservice)
```

See also

AddSOAPRequestHeader, AddSOAPResponseHeader, GetSOAPRequest, GetSOAPRequestHeader, GetSOAPResponseHeader, IsSOAPRequest; Basic web service concepts in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>webservice</td>
<td>A webservice object as returned from the cfobject tag or the CreateObject function.</td>
</tr>
</tbody>
</table>

Usage

Invoke the web service before attempting to get the response. You can use CFML XML functions to examine the XML response.

Example

This example makes a request to execute the echo_me function of the headerservice.cfc web service. Following the request, the example calls the GetSOAPResponse function to get the SOAP response, and then calls cfdump to display its content. For information on implementing the headerservice.cfc web service and also to see the echo_me function and the content of the web service CFC, see the example for either the AddSOAPResponseHeader function or the GetSOAPRequestHeader function.
<!--- Note that you might need to modify the URL in the CreateObject function to match your server and the location of the headerservice.cfc file if it is different than shown here. --->

<cfscript>
    ws = CreateObject("webservice",
    "http://localhost/soapheaders/headerservice.cfc?WSDL");
    ws.echo_me("hello world");
    resp = getSOAPResponse(ws);
</cfscript>
<cfdump var="#resp#"
GetSOAPResponseHeader

Description

Returns a SOAP response header. Call this function from within code that is invoking a web service after making a web service request.

Returns

A SOAP response header.

Category

XML functions

History

ColdFusion MX 7: Added this function.

Function syntax

```
GetSOAPResponseHeader(webservice, namespace, name [, asXML])
```

See also

AddSOAPRequestHeader, AddSOAPResponseHeader, GetSOAPRequest, GetSOAPRequestHeader, GetSOAPResponse, IsSOAPRequest; Basic web service concepts in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>webservice</td>
<td>A webservice object as returned from the cfobject tag or the CreateObject function.</td>
</tr>
<tr>
<td>namespace</td>
<td>A String that is the namespace for the header.</td>
</tr>
<tr>
<td>name</td>
<td>A String that is the name of the SOAP header.</td>
</tr>
<tr>
<td>asXML</td>
<td>If True, the header is returned as a CFML XML object. If False (default), the header is returned as a Java object.</td>
</tr>
</tbody>
</table>

Usage

If you specify false for the asXML parameter, ColdFusion first attempts to retrieve the header using the data type specified in the header’s type attribute. If the type attribute is not available, ColdFusion attempts to retrieve the header as a string. If you specify true for the asXML parameter, ColdFusion retrieves the header as raw XML. Used within CFML code by a consumer of a web service after it calls the web service with cfinvoke.

Example
There are two parts to this example. The first part is the web service CFC that this function (as well as the other ColdFusion SOAP functions) uses to demonstrate its interaction with a web service. To implement the web service for this function, see the example for either the AddSOAPResponseHeader function or the GetSOAPRequestHeader function. Execute the following example to see how the GetSOAPResponseHeader function operates:

```coldfusion
<!--- Note that you might need to modify the URL in the CreateObject function to match your server and the location of the headerservice.cfc file if it is different than shown here. Likewise for the cfinvoke tag at the end --->
<h3>GetSOAPResponseHeader Example</h3>
<cfscript>
// Create the web service object
ws = CreateObject("webservice", "http://localhost/soapheaders/headerservice.cfc?WSDL");

// Set the username header as a string
addSOAPRequestHeader(ws, "http://mynamespace/", "username", "tom", false);

// Set the password header as a CFML XML object
doc = XmlNew();
doc.password = XmlElemNew(doc, "http://mynamespace/", "password");
doc.password.XmlText = "My Voice is my Password";
doc.password.XmlAttributes["xsi:type"] = "xsd:string";
addSOAPRequestHeader(ws, "ignoredNameSpace", "ignoredName", doc);

// Invoke the web service operation
ret = ws.echo_me("argument");

// Get the first header as an object (string) and as XML
header = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader");
XMLheader = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader", true);

// Get the second header as an object (string) and as XML
header2 = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader2");
XMLheader2 = getSOAPResponseHeader(ws, "http://www.tomj.org/myns", "returnheader2", true);
</cfscript>
<br>
<cfoutput>
Soap Header value: #HTMLCodeFormat(header)#<br>
Soap Header XML value: #HTMLCodeFormat(XMLheader)#<br>
Soap Header 2 value: #HTMLCodeFormat(header2)#<br>
Soap Header 2 XML value: #HTMLCodeFormat(XMLheader2)#<br>
Return value: #HTMLCodeFormat(ret)#
</cfoutput>
<br>
<cfinvoke component="soapheaders.headerservice" method="echo_me" returnvariable="ret" in_here="hi"> </cfinvoke>
<cfoutput>Cfinvoke returned: #ret#</cfoutput>
```
GetSystemFreeMemory

Description

Gets details of free memory.

Returns

Memory, in bytes, that is currently free.

Syntax

getSystemFreeMemory()
GetSystemTotalMemory

Description

Gets details of the memory that is available for the operating system, in bytes.

Returns

Memory that is available in bytes.

Syntax

getSystemTotalMemory()
GetTempFile

Description

Creates a temporary file in a directory whose name starts with (at most) the first three characters of `prefix`.

Returns

Name of a temporary file, as a string.

Category

*System functions*

Function syntax

```
GetTempFile(dir, prefix)
```

See also

getTempDirectory

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dir</code></td>
<td>Directory name</td>
</tr>
<tr>
<td><code>prefix</code></td>
<td>Prefix of a temporary file to create in the <code>dir</code> directory</td>
</tr>
</tbody>
</table>

Example

```
<h3>GetTempFile Example</h3>
<p>The temporary directory for this ColdFusion Server is <cfoutput>#GetTempDirectory()#</cfoutput>.</p>
<p>We have created a temporary file called: <cfoutput>#GetTempFile(GetTempDirectory(),"testFile")#</cfoutput></p>
```
GetTemplatePath

Description

This function is deprecated. Use the GetBaseTemplatePath function instead. Gets the absolute path of an application’s base page.

History

ColdFusion MX: Deprecated this function. It might not work, and it might cause an error, in later releases.
GetTickCount

Description

Returns the current value of an internal millisecond timer.

Returns

A string representation of the system time, in milliseconds.

Category

Date and time functions, System functions

Function syntax

```coldfusion
GetTickCount()
```

Usage

This function is useful for timing CFML code segments or other page processing elements. The value of the counter has no meaning. To generate useful timing values, take the difference between the results of two GetTickCount calls.

Example

```coldfusion
<!---- Setup timing test ---->
<cfset iterationCount = 1000>
<!---- Time an empty loop with this many iterations ---->
<cfset tickBegin = GetTickCount()>
<cfloop Index = i From = 1 To = #iterationCount#></cfloop>
<cfset tickEnd = GetTickCount()>
<cfset loopTime = tickEnd - tickBegin>

<!---- Report ---->
<cfoutput>Loop time (#iterationCount# iterations) was: #loopTime# milliseconds</cfoutput>
```
GetTimeZoneInfo

Description

Gets local time zone information for the computer on which it is called, relative to Universal Time Coordinated (UTC). UTC is the mean solar time of the meridian of Greenwich, England, used as the basis for calculating standard time throughout the world. ColdFusion stores date and time values as date-time objects: real numbers on a universal time line. It converts an object to a time zone when it formats an object; it converts a date/time value from a time zone to a real number when it parses a value.

Returns

Structure that contains these elements and keys:

- utcTotalOffset: offset of local time, in seconds, from UTC
  - A plus sign indicates a time zone west of UTC (such as a zone in North America)
  - A minus sign indicates a time zone east of UTC (such as a zone in Germany)
- utcHourOffset: offset, in hours of local time, from UTC
- utcMinuteOffset: offset, in minutes, beyond the hours offset. For North America, this is 0. For countries that are not exactly on the hour offset, the number is from 0 through 60. For example, standard time in Adelaide, Australia is offset 9 hours and 30 minutes from UTC.
- isDSTOn: True, if Daylight Savings Time (DST) is on in the host; False, otherwise

Category

Date and time functions, International functions

Function syntax

```
GetTimeZoneInfo()
```

See also

DateConvert, CreateDateTime, DatePart

Example
<h3>GetTimeZoneInfo Example</h3>
<!--- This example shows the use of GetTimeZoneInfo --->
<cfoutput>
The local date and time are #now()#.  
</cfoutput>
<cfset info = GetTimeZoneInfo()>
<cfoutput>
Total offset in seconds is #info.utcTotalOffset#.  
Offset in hours is #info.utcHourOffset#.  
Offset in minutes minus the offset in hours is #info.utcMinuteOffset#.  
Is Daylight Savings Time in effect? #info.isDSTOn#.  
</cfoutput>
GetToken

Description

determines whether a token of the list in the delimiters parameter is present in a string.

Returns

The token found at position index of the string, as a string. If index is greater than the number of tokens in the string, returns an empty string.

Category

String functions

Function syntax

GetToken(string, index [, delimiters ])

See also

Left, Right, Mid, SpanExcluding, SpanIncluding

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String in which to search.</td>
</tr>
<tr>
<td>index</td>
<td>Positive integer or a variable that contains one. The position of a token.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. A delimited list of delimiters. Elements may consist of multiple characters. Default list of delimiters: space character, tab character, newline character; or their codes: &quot;chr(32)”, “chr(9)”, chr(10). Default list delimiter: comma character.</td>
</tr>
</tbody>
</table>

Usage

The following examples show how this function works.

Example 1

In the following example, the function call requests element number 2 from the string, using the delimiter ” [:; “.

GetToken("red,blue;red,black,tan;red,pink,brown;red,three", 2, " ; ")
The output is as follows:

```
red,black,tan
```

Example 2

```cfset mystring = "four,"
   & #chr(32) & #chr(9) & #chr(10) & ",five, nine,zero;"
   & #chr(10) & "nine, ten:, eleven:, twelve:, thirteen," & 
   & #chr(32) & #chr(9) & #chr(10) & 
   & ",four">
<cfoutput>
#HTMLCodeFormat(mystring)#<br><br>
</cfoutput>
```

The output is as follows:

```
four,
   ,five, nine,zero;,
nine, ten:, eleven:, twelve:, thirteen,
   ,four
```

The `getToken` function recognizes explicit spaces, tabs, or newline characters as the parameter delimiters. (To specify a space character, the code is `chr(32)`; a tab character, `chr(9)`; and a newline character, `chr(10)`.) In the example string `mystring`, there is:

- A forced space between the substrings "four," and ",five"
- A literal space between "five," and "nine"
- A literal space between "ten:," and "eleven,"
- A forced space between "thirteen," and ",four"

In the following call against `mystring`, no spaces are specified in `delimiters` (it is omitted), so the function uses the space character as the `string` delimiter:

```cfoutput>
  getToken(mystring, 3) is : #getToken(mystring, 3)#
</cfoutput><br>```

The output of this code is as follows:
GetToken(mystring, 3) is : nine,zero;:

The function finds the third delimiter, and returns the substring just before it that is between the second and third delimiter. This substring is ": nine,zero;:;\n
**Example 3**

```cfset
mystring2 = "four,"
 & #chr(9) & #chr(10) & ",five,nine,zero;:
 & #chr(10) & ",nine,ten:,eleven:,twelve;,thirteen," & #chr(9) & #chr(10) & ",four>
<cfoutput>
#HTMLCodeFormat(mystring2)#<br>
</cfoutput>
```

The output is as follows:

```
four,
, five, nine, zero;
nine, ten:, eleven:, twelve:, thirteen,
, four
```

The following is a call against `mystring2`:

```cfoutput
GetToken(mystring2, 2) is : #GetToken(mystring2, 2)#
</cfoutput>
```

The output is as follows:

```
GetToken(mystring2, 2) is :, five, nine, zero;
```

The function finds the second delimiter, and returns the substring just before it that is between the first and second delimiter. This substring is ": five, nine, zero;:;".

**Example**
<h3>GetToken Example</h3>
<cfif IsDefined("FORM.yourString")>
<!--- set delimiter --->
<cfif FORM.yourDelimiter is not ">"
    <cfset yourDelimiter = FORM.yourDelimiter>
<cfelse>
    <cfset yourDelimiter = " "></cfif>
</cfif>
<!---- check whether number of elements in list is greater than or equal to the element sought to return --->
<cfif ListLen(FORM.yourString, yourDelimiter) GTE FORM.returnElement>
<cfoutput>
<p>Element #FORM.ReturnElement# in #FORM.yourString#, delimited by 
"yourDelimiter"</p>
<br>is:#GetToken(FORM.yourString, FORM.returnElement, yourDelimiter)#
</cfoutput>
...</cfoutput>
GetTotalSpace

Description

Returns the total hard disk space or in-memory space available for an application.

Returns

Total space in bytes

Category

System functions

See also

getVFSMetadata

Syntax

gGetTotalSpace(path)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| path      | Path to the hard disk drive, for example C: for Windows or / for UNIX. Note that for Windows, you have to explicitly use colon (:) and not just C. Only the space of the entire hard disk drive is returned. If you mention a directory in the path, for example, /opt/, it is ignored and only {{/}} is considered. For in-memory file system, the path is ram:.

Usage

Application memory is determined by the following settings for RAM:

- In the ColdFusion Administrator, the value specified for Memory Limit per Application for In-Memory Virtual File System (Server Settings > Settings). The default value is 20 MB.
- The value specified in the Application.cfc for this.inmemoryfilesystem.size

If you have set values in both the places, the lesser value is considered.

⚠️ Note

Even if the value specified in Application.cfc is lesser, ensure that Server Settings > Settings > Enable In-Memory File System is checked in the ColdFusion Administrator.

Example
See the example for GetFreeSpace.
GetUserRoles

Description

Retrieves the list of roles for the current user. This function returns only ColdFusion roles, not roles set for the servlet API.

Returns

The list of roles for the current user.

Category

Security functions

Function syntax

```
GetUserRoles()
```

See also

cflogin, cfloginuser, cflogout, GetAuthUser, IsUserInAnyRole, IsUserInRole, IsUserLoggedIn, Securing Applications in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function.

Example

```
<cfloginuser name="#cflogin.name#" password="#cflogin.password#"
roles="#GetUserRoles()#" />
```
GetVFSMetaData

Description

Gets in-memory virtual file system metadata.

Returns

A structure that contains information about in-memory virtual file system.

Category

System functions

Function syntax

getVFSMetaData (fileSystemType)

See also

Working with in-memory files in the Developing ColdFusion Applications.

History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileSystemType</td>
<td>The in-memory file system type. ColdFusion 9 supports only RAM.</td>
</tr>
</tbody>
</table>

Usage

The function returns a structure with the following keys:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>If support for in-memory virtual file system is enabled.</td>
</tr>
<tr>
<td></td>
<td>This is the only key that is returned within the structure</td>
</tr>
<tr>
<td></td>
<td>if in-memory virtual file system is disabled.</td>
</tr>
<tr>
<td>Limit</td>
<td>The memory limit in bytes for in-memory virtual file system.</td>
</tr>
<tr>
<td>Used</td>
<td>The memory out of the specified memory limit that is in use (in bytes).</td>
</tr>
<tr>
<td>Free</td>
<td>The free memory in bytes.</td>
</tr>
</tbody>
</table>

Example
<cfset myroot = hash(getDirectoryFromPath(getCurrentTemplatePath())) >
<cfset name = "ram:///&myroot&/">
<cffile action="append" file="#name#" output="created at #now()#">
<cfset contents = fileRead(name)>
<cfdump var="#contents#">
<cfdump var="#getVFSMetaData("ram")#">

The example works only if in-memory virtual file system is enabled.
GetWriteableImageFormats

Description

Returns a list of image formats that ColdFusion can write on the operating system where ColdFusion is deployed.

Returns

A list of image file formats.

Category

System functions

History

ColdFusion 8: Added this function.

Function syntax

GetWriteableImageFormats()

See also

GetReadableImageFormats, cfimage (for supported image file formats)

Usage

Use this function to determine image file compatibility on the ColdFusion server.

Example

<cfoutput>#GetWriteableImageFormats()#</cfoutput>

#back to top
GetSafeHTML

getSafeHTML() sanitizes the given input string based on the rules specified in an antisamy policy file. This can be used to prevent unwanted user supplied HTML being used in an application.

*Returns*

Returns the sanitized input. If throwOnError is set to true, it returns an error. Otherwise, it returns an empty string.

*Returns*

ColdFusion 11: Added this function

*See Also*

isSafeHTML

*Parameter*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to be encoded.</td>
</tr>
<tr>
<td>PolicyFile</td>
<td>Optional. File path for antisamy policy file which specifies HTML rules. In case if not specified, there is a provision to set this at application level. Else the default policy file shipped with ColdFusion will be used.</td>
</tr>
<tr>
<td>throwOnError</td>
<td>Optional. If set to true, exception is thrown.</td>
</tr>
</tbody>
</table>

*Example*

The following example illustrates a sample Antisamy component:

```coldfusion
<cfcomponent>
  <cfset this.security.antisamypolicy = "antisamy.xml">
</cfcomponent>
```

The following example illustrates isSafeHTML and getSafeHTML validation.

In the example, the input HTML file is a GIF image. On processing it through the functions (isSafeHTML, getSafeHTML), checks if the given input string violates the rules specified in antiSamy policy file and returns true if the input does not violate the allowed HTML rules.

```coldfusion
<cfset  isSafe = isSafeHTML(inputHTML)>
<cfset  SafeHTML = getSafeHTML(inputHTML, ",", true)>
<cfoutput> is Safe : #isSafe#  Safe HTML : #SafeHTML#  </cfoutput>
```
EncodeForXMLAttribute

Description

Encodes a string that can be used as an XML attribute.

Returns

Encoded XML string.

Category

Display and formatting functions.

Function syntax

```
encodeForXMLAttribute(String inputString, boolean canonicalize)
```

See also

Canonicalize, EncodeForHTMLAttribute, EncodeForJavaScript, EncodeForCSS, EncodeForURL

History

ColdFusion 11: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to encode.</td>
</tr>
<tr>
<td>canonicalize</td>
<td>Optional. If set to true, canonicalization happens before encoding. If set to false, the given input string will just be encoded. The default value for canonicalize is false. When this parameter is not specified, canonicalization will not happen. By default, when canonicalization is performed, both mixed and multiple encodings will be allowed. To use any other combinations you should canonicalize using canonicalize method and then do encoding.</td>
</tr>
</tbody>
</table>

Example
<!-- Prints ,.-_ ---->
<cfset x=encodeForXMLAttribute("',.-_")>
<cfoutput>#x#</cfoutput>

<!-- Prints &amp;#x3c;script&amp;#x3e; ---->
<cfset x=encodeForXMLAttribute("&lt;script&gt;")>
<cfoutput>#x#</cfoutput>

<!-- Prints !@$%()=+{[} ---->
<cfset x=encodeForXMLAttribute("!@$%()=+{["")>
<cfoutput>#x#</cfoutput>

<!-- Prints &amp;#x21;&amp;#x40;&amp;#x24;&amp;#x25;&amp;#x28;&amp;#x29;&amp;#x3d;&amp;#x2b;&amp;#x7b;&amp;#x7d;&amp;#x5b;&amp;#x5d; ---->
<cfset x=encodeForXMLAttribute("!@$%()=+{["")>
<cfoutput>#x#</cfoutput>
EncodeForXPath

Description

Returns an encoded string for using with XPATH.

Returns

Encoded XML string.

Category

Display and formatting functions.

Function syntax

```plaintext
encodeForXPath(String inputString, boolean canonicalize)
```

See also

Canonicalize, EncodeForHTMLAttribute, EncodeForJavaScript, EncodeForCSS, EncodeForURL

History

ColdFusion 11: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to encode.</td>
</tr>
<tr>
<td>canonicalize</td>
<td>Optional. If set to true, canonicalization happens before encoding. If set to false, the given input string will just be encoded. The default value for canonicalize is false. When this parameter is not specified, canonicalization will not happen. By default, when canonicalization is performed, both mixed and multiple encodings will be allowed. To use any other combinations you should canonicalize using canonicalize method and then do encoding.</td>
</tr>
</tbody>
</table>

Example
<!---- Prints or 1---->

<cfset x=encodeForXPath("or 1=1",false)>
<cfoutput>#x#</cfoutput>
Functions h-im

- Hash
- HMAC
- HQLMethods
- Hour
- HTMLCodeFormat
- HTMLEditFormat
- If
- ImageAddBorder
- ImageBlur
- ImageClearRect
- ImageCopy
- ImageCreateCaptcha
- ImageCrop
- ImageDrawArc
- ImageDrawBeveledRect
- ImageDrawCubicCurve
- ImageDrawLine
- ImageDrawLines
- ImageDrawOval
- ImageDrawPoint
- ImageDrawQuadraticCurve
- ImageDrawRect
- ImageDrawRoundRect
- ImageDrawText
- ImageFlip
- ImageGetBlob
- ImageGetBufferedImage
- ImageGetEXIFMetadata
- ImageGetEXIFTag
- ImageGetHeight
- ImageGetIPTCMetadata
- ImageGetIPTCTag
- ImageGetWidth
- ImageGrayscale
- ImageInfo
- ImageMakeColorTransparent
- ImageMakeTranslucent
- ImageNegative
- ImageNew
- ImageOverlay
- ImagePaste
- ImageRead
- ImageReadBase64
- ImageResize
- ImageRotate
- ImageRotateDrawingAxis
- ImageScaleToFit
- ImageSetAntialiasing
- ImageSetBackgroundColor
- ImageSetDrawingColor
- ImageSetDrawingStroke
- ImageSetDrawingTransparency
- ImageSharpen
- ImageShear
- ImageShearDrawingAxis
- ImageTranslate
- ImageTranslateDrawingAxis
- ImageWrite
- ImageWriteBase64
- ImageXORDrawingMode
- ImageGetMetadata
Hash

Description

Converts a variable-length string to a fixed-length string that can act as a "fingerprint" or identifier for the original string. It is not possible to convert the hash result back to the source string.

Returns

A string.

Category

Conversion functions, Security functions, String functions

Function syntax

Hash(string [, algorithm [, encoding] [, iterations ]])

History

ColdFusion MX 7: Added the algorithm and encoding parameters.

ColdFusion 10: Added the iterations argument.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>String to hash.</td>
</tr>
</tbody>
</table>
### algorithm

(Conditional) The algorithm to use to hash the string. ColdFusion installs a cryptography library with the following algorithms:

- **CFMX_COMPAT**: Generates a hash string identical to that generated by ColdFusion MX and ColdFusion MX 6.1 (default). It is only a placeholder algorithm that informs ColdFusion to use an algorithm compatible with CFMX if the user does not have any option to provide an algorithm.
- **MD5**: (default) Generates a 32-character, hexadecimal string, using the MD5 algorithm (The algorithm used in ColdFusion MX and prior releases).
- **SHA**: Generates a 40-character string using the Secure Hash Standard SHA-1 algorithm specified by Nation Institute of Standards and Technology (NIST) FIPS-180-2.
- **SHA-256**: Generates a 44-character string using the SHA-256 algorithm specified by FIPS-180-2.
- **SHA-384**: Generates a 64-character string using the SHA-384 algorithm specified by FIPS-180-2.
- **SHA-512**: Generates an 128-character string using the SHA-1 algorithm specified by FIPS-180-2.

The Enterprise Edition of ColdFusion installs the RSA BSafe Crypto-J library, which provides FIPS-140 Compliant Strong Cryptography. It includes the following algorithms:

- **MD2**: The MD2 hash algorithm defined by RFC 1319.
- **MD5**: The defined by RFC 1321.
- **RIPEMD160**: RACE Integrity Primitives Evaluation Message Digest 160-bit message digest algorithm and cryptographic hash function.
- **SHA-1**: The 160-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198.
- **SHA-224**: The 224-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198.
- **SHA-256**: The 256-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198.
- **SHA-384**: The 384-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198.
- **SHA-512**: The 512-bit secure hash algorithm defined by FIPS 180-2 and FIPS 198.

If you install a security provider with additional cryptography algorithms, you can also specify any of its hashing algorithms.
encoding (Optional; to use this attribute, also specify the algorithm parameter) A string specifying the encoding to use when converting the string to byte data used by the hash algorithm. Must be a character encoding name recognized by the Java runtime. The default value is the value specified by the defaultCharset entry in the neo-runtime.xml file, which is normally UTF-8. Ignored when using the CFMX_COMPAT algorithm.

iterations (Optional) The number of times to iterate hashing, to increase hash computation time. CF10+

Usage

The result of this function is useful for comparison and validation. For example, you can store the hash of a password in a database without exposing the password. You can check the validity of the password by hashing the entered password and comparing the result with the hashed password in the database. ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes facilities for using other provider implementations; however, Adobe cannot provide technical support for third-party security providers. The encoding attribute is normally not required. It provides a mechanism for generating identical hash values on systems with different default encodings. ColdFusion uses a default encoding of UTF-8 unless you modify the defaultCharset entry in the neo-runtime.xml file.

Example

The following example lets you enter a password and compares the hashed password with a hash value saved in the SecureData table of the cfdocexamples database. This table has the following entries:

<table>
<thead>
<tr>
<th>User ID</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>blaw</td>
<td>blaw</td>
</tr>
<tr>
<td>dknob</td>
<td>dknob</td>
</tr>
</tbody>
</table>
<h3>Hash Example</h3>

<!--- Do the following if the form is submitted. --->
<cfif IsDefined("Form.UserID")>

<!--- query the data base. --->
<cfquery name = "CheckPerson" datasource = "cfdocexamples">
SELECT PasswordHash
FROM SecureData
WHERE UserID = <cfqueryparam value = "#Form.userID#" cfsqltype = 'CF_SQL_VARCHAR'></cfquery>

<!--- Compare query PasswordHash field and the hashed form password and display the results. --->
<cfoutput>
<cfif Hash(Form.password, "SHA") is not checkperson.passwordHash>
User ID #Form.userID# or password is not valid. Try again.
<cfelse>
Password is valid for User ID #Form.userID#.
</cfif>
</cfoutput>
</cfif>

<!--- Form for entering ID and password. --->
<form action="#CGI.SCRIPT_NAME#" method="post">
<b>User ID: </b>
<input type = "text" name="UserID" ><br>
<b>Password: </b>
<input type = "text" name="password" ><br><br>
<input type = "Submit" value = "Encrypt my String">
</form>
HMac

Description

Creates Hash-based Message Authentication Code for the given string based on the algorithm and encoding. Hash-based Message Authentication Code (HMAC) is used to verify the data integrity and authenticity of a message transmitted. It involves a cryptographic hash function in combination with a secret key. The cryptographic hash function can be Message Digest 5 (MD5), Secure Hash Algorithm (SHA), and so on.

Returns

An encoded string

Category

Display and formatting functions

History

ColdFusion 10: Added this function.

Syntax

HMac(message, key [,algorithm] [,encoding])

See also

SessionInvalidate, SessionRotate

History

ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required\Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>Required</td>
<td>The message to transmit. The message can be a String or a byte array.</td>
</tr>
<tr>
<td>key</td>
<td>Required</td>
<td>The secret key to create HMAC. The key can be a String or a byte array.</td>
</tr>
<tr>
<td>algorithm</td>
<td>Optional</td>
<td>Algorithm used.</td>
</tr>
<tr>
<td>encoding</td>
<td>Optional</td>
<td>Encoding to be used.</td>
</tr>
</tbody>
</table>

Usage

Use this function to create Hash-based Message Authentication Code for the given string based on the algorithm and encoding.

Example
<h2>HMAC Test</h2>
<cfset x=hmac("Hi There","key1","HMACRIPEMD160")>
<cfoutput>#x#</cfoutput>
HQLMethods

The Hibernate Query Language (HQL) methods return a single or multi-dimensional array of values or entities, based on what the HQL query returns. If you are sure that only one record exists that matches this filter criteria, specify `unique=true` so that a single entity is returned instead of an array. If `unique=true` and multiple records are returned, then an exception is thrown.

**Note**

*entityname* and *properties* used in HQL are case sensitive.

The following HQL methods are available:

- `ORMExecuteQuery(hql, [unique], [queryoptions])`
- `ORMExecuteQuery(hql, params [unique] [queryOptions])`
- `ORMExecuteQuery(hql, namedparams [unique] [queryOptions])`

### ORMExecuteQuery(hql, [unique] [queryoptions])

**Description**

Runs HQL on the default data source specified for the application. You can specify several options to control the behavior of retrieval using `queryOptions`:

- `ignorecase`: Ignores the case of sort order when you set it to true. Use this option only when you specify the `sortorder` parameter.
- `maxResults`: Specifies the maximum number of objects to be retrieved.
- `offset`: Specifies the start index of the resultset from where it has to start the retrieval.
- `cacheable`: Whether the result of this query is to be cached in the secondary cache. Default is false.
- `cachename`: Name of the cache in secondary cache.
- `timeout`: Specifies the timeout value (in seconds) for the query

**Note**

Maxresults and timeout are used for pagination.

**Category**

ORM functions

**Example**

```cfc
<cfset employees = ORMExecuteQuery("from Employees")>
<cfset employees = ORMExecuteQuery("from Employees where age > 40")>
<cfset employeeObj = ORMExecuteQuery("from Employees where EmployeeID =1", true)>
<cfset firstNameArray = ORMExecuteQuery("select FirstName from Employees")>
<cfset numberOfEmps = ORMExecuteQuery("select count(*) from Employees")>
<cfset firstName = ORMExecuteQuery("select FirstName from Employees where EmployeeID = 1", true)>
<cfset employees = ORMExecuteQuery("from Employees", false, {offset=5, maxresults=10, timeout=5})>
```
Description

This type of ORMExecuteQuery lets you pass parameters to the query. Use '?' (question mark) as the placeholder for the parameters. The values to the parameters must be passed as an array to params. In addition, you can specify several options to control the behavior of retrieval using queryOptions:

- **ignorecase**: Ignores the case of sort order when you set it to true. Use this option only when you specify the sortorder parameter.
- **maxResults**: Specifies the maximum number of objects to be retrieved.
- **offset**: Specifies the start index of the resultset from where it has to start the retrieval.
- **cacheable**: Whether the result of this query is to be cached in the secondary cache. Default is false.
- **cachename**: Name of the cache in secondary cache.
- **timeout**: Specifies the timeout value (in seconds) for the query.

**Note**

Maxresults and timeout are used for pagination.

Category

ORM functions

Example

```cfscript
<cfset employees = ORMExecuteQuery("from Employee where age > ?", [40])>
<cfset employeeObj = ORMExecuteQuery("from Employee where EmployeeID=?", [1], true)>
<cfset employees = ORMExecuteQuery("from Employee where age > ? and age < ?", [40, 80])>
```

ORMExecuteQuery(hql, namedparams [, unique] [, queryOptions])

Description

This type of ORMExecuteQuery lets you pass named parameters to the query. The placeholder for the parameter must be a name and must start with ":" as in ":age" or ":id". The values to the names must be passed as key-value pairs. In addition, you can specify several options to control the behavior of retrieval using queryOptions:

- **ignorecase**: Ignores the case of sort order when you set it to true. Use this option only when you specify the sortorder parameter.
- **maxResults**: Specifies the maximum number of objects to be retrieved.
- **offset**: Specifies the start index of the resultset from where it has to start the retrieval.
- **cacheable**: Whether the result of this query is to be cached in the secondary cache. Default is false.
- **cachename**: Name of the cache in secondary cache.
- **timeout**: Specifies the timeout value (in seconds) for the query.

**Note**

Maxresults and timeout are used for pagination.
ORM functions

Example

To retrieve employee details of all employees whose reside in USA and are also citizens of USA:

<cfset USEmployees = ORMExecuteQuery("from Employee where country=:country and citizenship=:country", {country='USA'})>
<cfset orderDetail = ORMExecuteQuery("from Orders where OrderID=:orderid and ProductID=:productid", {orderid=1, productid=901}, true)>
### Hour

**Description**

Gets the current hour of the day.

**Returns**

Ordinal value of the hour, in the range 0-23.

**Category**

**Date and time functions**

**Function syntax**

```
Hour(date)
```

**See also**

**DatePart, Minute, Second**

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

**Usage**

When passing a date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

**Example**

```
<!--- This example shows the use of Hour, Minute, and Second --->
<h3>Hour Example</h3>
<cfoutput>
The time is currently #TimeFormat(Now())#.
We are in hour #Hour(Now())#, Minute #Minute(Now())# and Second #Second(Now())# of the day.
</cfoutput>
```
**HTMLCodeFormat**

**Description**

Replaces special characters in a string with their HTML-escaped equivalents and inserts `<pre>` and `</pre>` tags at the beginning and end of the string.

**Returns**

HTML-escaped string *string*, enclosed in `<pre>` and `</pre>` tags. Return characters are removed; line feed characters are preserved. Characters with special meanings in HTML are converted to HTML character entities such as `>`. 

**Category**

*Display and formatting functions*

**Function syntax**

```
HTMLCodeFormat(string [, version ])
```

**See also**

*HTMLEditFormat*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one.</td>
</tr>
<tr>
<td>version</td>
<td>HTML version to use; currently ignored.</td>
</tr>
<tr>
<td></td>
<td>• -1: The latest implementation of HTML</td>
</tr>
<tr>
<td></td>
<td>• 2.0: HTML 2.0 (default)</td>
</tr>
<tr>
<td></td>
<td>• 3.2: HTML 3.2</td>
</tr>
</tbody>
</table>

**Usage**

This function converts the following characters to HTML character entities:

<table>
<thead>
<tr>
<th>Text character</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;</code></td>
<td><code>&lt;</code></td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td><code>&gt;</code></td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td><code>&amp;</code></td>
</tr>
<tr>
<td><code>&quot;</code></td>
<td><code>&quot;</code></td>
</tr>
</tbody>
</table>
This function typically increases the length of a string. This can cause unpredictable results when performing certain string functions (Left, Right, and Mid, for example) against the expanded string. The only difference between this function and HTMLEditFormat is that HTMLEditFormat does not surround the text in an HTML pre tag.

Example

<!--- This example shows the effects of HTMLCodeFormat and HTMLEditFormat. View it in your browser; then View it using your browser’s the View Source command. --->
<cfset testString="This is a test & this is another
This text is in angle brackets">

Previous line was blank!!!

<cfoutput>
<h3>The text without processing</h3>
#testString#<br>
<h3>Using HTMLCodeFormat</h3>
#HTMLCodeFormat(testString)#
<h3>Using HTMLEditFormat</h3>
#HTMLEditFormat(testString)#
</cfoutput>
HTMLEditFormat

Adobe recommends that you use the EncodeForHTML function, not the HTMLEditFormat function, to escape special characters in a string for use in HTML in all new applications.

THIS FUNCTION HAS BEEN DEPRECATED

Description

Replaces special characters in a string with their HTML-escaped equivalents.

Returns

HTML-escaped string string. Return characters are removed; line feed characters are preserved. Characters with special meanings in HTML are converted to HTML character entities such as >.

Category

Display and formatting functions

Function syntax

HTMLEditFormat(string [, version ])

See also

HTMLEscapeFormat, cfapplication

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one.</td>
</tr>
<tr>
<td>version</td>
<td>HTML version to use; currently ignored.</td>
</tr>
</tbody>
</table>

-1: The latest implementation of HTML
2.0: HTML 2.0 (default)
3.2: HTML 3.2

Usage

This function converts the following characters to HTML character entities:

<table>
<thead>
<tr>
<th>Text character</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
</tbody>
</table>
This function can be used to help protect ColdFusion pages that return user-provided data to the client browser from cross-site scripting attacks. However, the `scriptprotect` attribute of the `cfapplication` tag or the equivalent `This.scriptProtect` variable setting in Application.cfc can be preferable in most instances, because you only need to specify it once for an application. This function typically increases the length of a string. This can cause unpredictable results when performing certain string functions (`Left`, `Right`, and `Mid`, for example) against the expanded string. The only difference between this function and `HTMLCodeFormat` is that `HTMLCodeFormat` surrounds the text in an HTML `pre` tag.

Example

```cfml
<!--- This example shows the effects of HTMLCodeFormat and HTMLEditFormat. View it in your browser, then View it using your browser's the View Source command. --->
<cfset testString="This is a test
& this is another
<This text is in angle brackets>
Previous line was blank!!!">

<cfoutput>
<h3>The text without processing</h3>
#testString#
<br>
<h3>Using HTMLCodeFormat</h3>
#HTMLCodeFormat(testString)#
<h3>Using HTMLEditFormat</h3>
#HTMLEditFormat(testString)#
</cfoutput>
```
IIf

Description

Evaluates a Boolean conditional dynamic expression. Depending on whether the expression is yes or no, dynamically evaluates one of two string expressions and returns the result. This function is convenient for incorporating a cfif tag in-line in HTML. For general conditional processing, see cfif. For error handling, see cftry. For more information, see the Developing ColdFusion Applications.

Returns

If result is yes, returns the value of Evaluate(string_expression1); otherwise, returns the value of Evaluate(string_expression2).

Category

Decision functions, Dynamic evaluation functions

Function syntax

```
IIf(condition, string_expression1, string_expression2)
```

See also

DE, Evaluate

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>An expression that can be evaluated as a Boolean.</td>
</tr>
<tr>
<td>string_expression1</td>
<td>A string or a variable that contains one. Expression to evaluate and return if condition is yes.</td>
</tr>
<tr>
<td>string_expression2</td>
<td>A string or a variable that contains one. Expression to evaluate and return if condition is no.</td>
</tr>
</tbody>
</table>

Usage

The IIf function is a shortcut for the following construct:

```
<cfif condition>
    <cfset result = Evaluate(string_expression1)>
<cfelse>
    <cfset result = Evaluate(string_expression2)>
</cfif>
```

The expressions string_expression1 and string_expression2 must be string expressions, so that they are not evaluated immediately as the parameters of IIf. For example:
If \( y = 0 \), this generates an error, because the third expression is the value of \( x/0 \) (invalid expression). ColdFusion evaluates string_expression1 and string_expression2. To return the string itself, use the `DE` function.

**Note**

If you use number signs (#) in string_expression1 or string_expression2, ColdFusion evaluates the part of the expression in number signs first. If you misuse the number signs, you can cause unexpected results from the IIf function. For example, if you use number signs around the whole expression in string_expression1, and if there is an undefined variable in string_expression1, the function might fail, with the error “Error Resolving Parameter.”

If a variable is undefined, ColdFusion throws an error when it processes this function. The following example shows this problem:

```cfc
#IIf(IsDefined("Form.Deliver"), DE(Form.Deliver), DE("no"))#
```

This returns “Error resolving parameter FORM.DELIVER”. To avoid this problem, use the `DE` and `Evaluate` functions in code such as the following:

```cfc
#IIf(IsDefined("Form.Deliver"), Evaluate(DE("Form.Deliver")), DE("no"))#
```

This returns “no”; ColdFusion does not throw an error. In the following example, LocalVar is undefined; however, if you omit number signs around LocalVar, the code works properly:

```cfc
<cfoutput>
#IIf(IsDefined("LocalVar"), "LocalVar",
    DE("The variable is not defined.")#
</cfoutput>
```

The output is:

```
The variable is not defined.
```

The number signs around LocalVar in the following code cause it to fail with the error message ‘Error Resolving Parameter’, because ColdFusion never evaluates the original condition `IsDefined("LocalVar")`. Here is another example:
The error message would be as follows:

```
Error resolving parameter LOCALVAR
```

The `DE` function has no effect on the evaluation of `LocalVar`, because the number signs cause it to be evaluated immediately.

Example

```
<IIf Evaluates a condition, and does an Evaluate on string
expression 1 or string expression 2 depending on the Boolean
outcome <I>(yes: run expression 1; no: run expression 2)</I>.</p>

The result of the expression

```
IIf( Hour(Now()) GTE 12,
   DE("It is afternoon or evening"),
   DE("It is morning"))
```

is:

```
<cfoutput>
#IIf( Hour(Now()) GTE 12,
   DE("It is afternoon or evening"),
   DE("It is morning"))#
</cfoutput>
```
ImageAddBorder

Description

Adds a rectangular border around the outside edge of a ColdFusion image.

Returns

Nothing.

Category

Image functions

Function syntax

ImageAddBorder(name, thickness [, color, borderType])

See also

cfimage, ImageDrawRect, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>thickness</td>
<td>Required. Thickness of the border in pixels. The default value is 1. The border is added to the outside edge of the image; the image area is increased accordingly.</td>
</tr>
<tr>
<td>color</td>
<td>Optional. Border color. The default border color is black. See Usage. Only valid if the borderType is not specified or if borderType = &quot;constant&quot;.</td>
</tr>
</tbody>
</table>
borderType

<table>
<thead>
<tr>
<th>Optional. The type of border:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>zero</strong>: Sets the border color to black.</td>
</tr>
<tr>
<td><strong>constant</strong>: Sets the border to the specified color (default).</td>
</tr>
<tr>
<td><strong>copy</strong>: Sets sample values to copies of the nearest valid pixel. For example, pixels to the left of the valid rectangle assume the value of the valid edge pixel in the same row. Pixels both above and to the left of the valid rectangle assume the value of the upper-left pixel.</td>
</tr>
<tr>
<td><strong>reflect</strong>: Mirrors the edges of the source image. For example, if the left edge of the valid rectangle is located at ( x = 10 ), pixel ((9, y)) is a copy of pixel ((10, y)) and pixel ((6, y)) is a copy of pixel ((13, y)).</td>
</tr>
<tr>
<td><strong>wrap</strong>: Tiles the source image in the plane.</td>
</tr>
</tbody>
</table>

Usage

Thickness of the border in pixels. The default value is 1. The border is added to the outside edge of the image; the image area is increased accordingly.

Example

Example 1

```cfml
<!--- This example shows how to create a 10-pixel-wide red border around an image with a 5-pixel-wide green border around the red border. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Draw a red border around the outside edge of the image. --->
<cfset ImageAddBorder(myImage,10,"red")>
<!--- Draw a green border around the outside edge of the red border. --->
<cfset ImageAddBorder(myImage,5,"green")>
<!--- Save the modified ColdFusion image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpeg"/>
```

Example 2
Example 3

<!---- This example shows how to create a 100-pixel-wide border that is a mirror of the source image. --->
<!---- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/maxwell01.jpg" name="myImage">
<!---- Create the border. --->
<cfset ImageAddBorder(myImage,100,"","reflect")>
<!---- Save the modified ColdFusion image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!---- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/maxwell01.jpg"/>
<img src="test_myImage.jpeg"/>

Example 4

<!---- This example shows how to copy 100 pixels from the outer edge of the image and create a border from it. --->
<!---- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<cfset ImageAddBorder(myImage,100,"","copy")>
<!---- Save the modified ColdFusion image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!---- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpeg"/>
ImageBlur

Description
Smooths (blurs) the ColdFusion image.

Returns
Nothing.

Category
Image functions

Function syntax

```plaintext
ImageBlur(name [, blurRadius])
```

See also
ImageSharpen, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>blurRadius</td>
<td>Optional. The size of the blur radius. Value must be greater than or equal to 3 and less than or equal to 10. The default value is 3.</td>
</tr>
</tbody>
</table>

Usage

The `blurRadius` operation affects performance: as the `blurRadius` value increases, performance decreases.

Example

```coldfusion
```

```
```

<!--- This example shows how to blur an image by a radius of 10. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfset ImageBlur(myImage,10)>
<!--- Use the maximum blur radius to blur the image. --->
<!--- Save the modified ColdFusion image to a JPEG file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpeg"/>
**ImageClearRect**

**Description**

Clears the specified rectangle by filling it with the background color of the current drawing surface.

**Returns**

Nothing.

**Category**

Image functions

**Function syntax**

```
ImageClearRect(name, x, y, width, height)
```

**See also**

*ImageSetBackgroundColor*, *IsImageFile*

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate of the rectangle to clear.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate of the rectangle to clear.</td>
</tr>
<tr>
<td>width</td>
<td>Required. The width of the rectangle to clear.</td>
</tr>
<tr>
<td>height</td>
<td>Required. The height of the rectangle to clear.</td>
</tr>
</tbody>
</table>

**Usage**

Use this function in conjunction with the `ImageSetBackgroundColor` function.

**Example**


<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Set the background color to green. --->
<cfset ImageSetBackgroundColor(myImage,"green")>
<!--- Draw four rectangles in the background color. --->
<cfset ImageClearRect(myImage,10,25,50,50)>
<cfset ImageClearRect(myImage,100,25,50,50)>
<cfset ImageClearRect(myImage,10,100,50,50)>
<cfset ImageClearRect(myImage,100,100,50,50)>
<!--- Save the modified ColdFusion image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpeg"/>
ImageCopy

Description

Copies a rectangular area of an image.

Returns

A ColdFusion image for the copied area.

Category

Image functions

Function syntax

ImageCopy(name, x, y, width, height [, dx, dy])

See also

ImageNew, ImagePaste, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate of the source rectangle.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate of the source rectangle.</td>
</tr>
<tr>
<td>width</td>
<td>Required. The width of the source rectangle.</td>
</tr>
<tr>
<td>height</td>
<td>Required. The height of the source rectangle.</td>
</tr>
<tr>
<td>dx</td>
<td>Optional. The x coordinate of the destination rectangle.</td>
</tr>
<tr>
<td>dy</td>
<td>Optional. The y coordinate of the destination rectangle.</td>
</tr>
</tbody>
</table>

Usage

The rectangle is specified by \((x, y, width, height)\). The area is copied to the rectangle with the upper-left corner specified by \((dx, dy)\).

Example
Example 1

<!--- This example shows how to copy a rectangular area of an image to a new image. --->

<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/lori05.jpg" name="myImage">

<!--- Copy the rectangular area specified by the coordinates (25,25,50,50) in the image to the rectangle beginning at (75,75), and return this copied rectangle as a new ColdFusion image. --->
<cfset dupArea = ImageCopy(myImage,25,25,50,50,75,75)>

<!--- Write the result to a PNG file. --->
<cfimage source=#myImage# action="write" destination="test_myImage.png" overwrite="yes">

<!--- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/lori05.jpg">
<img src="test_myImage.png">

Example 2

<!--- This example shows how to copy a rectangular area from one image and paste it over another image. --->

<!--- Create a ColdFusion image named "myImage1" from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/lori05.jpg" name="myImage1">

<!--- Create the ColdFusion image "myImage2" from a different JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/maxwell01.jpg" name="myImage2">

<!--- Copy a rectangular region of "myImage1" as a new image. --->
<cfset resImage = ImageCopy(myImage1,1,1,55,55)>

<!--- Paste the rectangular area on the second image. --->
<cfset ImagePaste(myImage2,resImage,50,75)>

<!--- Write the second ColdFusion image to a file. --->
<cfimage action="write" source="#myImage2#" destination="test_myImage.jpg" overwrite="yes">

<!--- Display the two source files and the new file. --->
<img src="../cfdocs/images/artgallery/lori05.jpg"/>
<img src="../cfdocs/images/artgallery/maxwell01.jpg"/>
<img src="test_myImage.jpg"/>
ImageCreateCaptcha

Description

Create a Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) image, a distorted text image that is human-readable, but not machine-readable, used in a challenge-response test for preventing spam.

Returns

Image object

History

ColdFusion 10: Added this function.

Syntax

imageCreateCaptcha (height, width, text)imageCreateCaptcha (height, width, text [, difficulty])imageCreateCaptcha (height, width, text [, difficulty, fonts, fontsize)

See also
cfimage

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>Required. Height in pixels of the image.</td>
</tr>
<tr>
<td>width</td>
<td>Required. Width in pixels of the image.</td>
</tr>
<tr>
<td>text</td>
<td>Required. Text string displayed in the CAPTCHA image. Use capital letters for better readability. Do not include spaces because users cannot detect them in the resulting CAPTCHA image.</td>
</tr>
<tr>
<td>difficulty</td>
<td>Optional. Level of complexity of the CAPTCHA text. Specify one of the following levels of text distortion:</td>
</tr>
<tr>
<td></td>
<td>low</td>
</tr>
<tr>
<td></td>
<td>medium</td>
</tr>
<tr>
<td></td>
<td>high</td>
</tr>
<tr>
<td>font</td>
<td>Optional. One or more valid fonts to use for the CAPTCHA text. Separate multiple fonts with commas. ColdFusion supports only the system fonts that the JDK can recognize. For example, TTF fonts in the Windows directory are supported on Windows.</td>
</tr>
</tbody>
</table>
**fontsize**

Optional. Font size of the text in the CAPTCHA image. The value must be an integer.

**Example**

```cfc
<h1>ImageCreateCaptcha Method</h1>
<cfset funcimg1 = ImageCreateCaptcha(35,400,"loner")>
   <cfimage action="writetoBrowser" source="#funcimg1#">
<cfset funcimg2 = ImageCreateCaptcha(35,400,"loner","high")>
   <cfimage action="writetoBrowser" source="#funcimg2#">
<cfset funcimg3 = ImageCreateCaptcha(35,400,"loner","high","serif,sansserif","24")>
   <cfimage action="writetoBrowser" source="#funcimg3#">
```
ImageCrop

Description
Crops a ColdFusion image to a specified rectangular area.

Returns
Nothing.

Category
Image functions

Function syntax

```
ImageCrop(name, x, y, width, height)
```

See also
ImageFlip, ImageResize, IsImageFile

History
ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x origin of the crop area.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y origin of the crop area.</td>
</tr>
<tr>
<td>width</td>
<td>Required. The width of the crop area.</td>
</tr>
<tr>
<td>height</td>
<td>Required. The height of the crop area.</td>
</tr>
</tbody>
</table>

Usage
The rectangular area cannot be empty, and must be fully contained within the source image bounds.

Example
<!--- This example shows how to crop an image. --->
<!---- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/lori05.jpg" name="myImage">
<!---- Crop myImage to 100x100 pixels starting at the coordinates (10,10). --->
<cfset ImageCrop(myImage,10,10,100,100)>
<!---- Write the result to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpg"
overwrite="yes">
<!---- Display the source image and the new image. --->
<img src="../cfdocs/images/artgallery/lori05.jpg"/>
<img src="test_myImage.jpg"/>
ImageDrawArc

Description

Draws a circular or elliptical arc.

Returns

Nothing.

Category

Image functions

Function syntax

ImageDrawArc(name, x, y, width, height, startAngle, arcAngle [, filled])

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate of the upper-left corner of the arc.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate of the upper-left corner of the arc.</td>
</tr>
<tr>
<td>width</td>
<td>Required. The width of the arc.</td>
</tr>
<tr>
<td>height</td>
<td>Required. The height of the arc.</td>
</tr>
<tr>
<td>startAngle</td>
<td>Required. The beginning angle in degrees.</td>
</tr>
<tr>
<td>arcAngle</td>
<td>Required. The angular extent of the arc, relative to the start angle.</td>
</tr>
</tbody>
</table>
Optional. Specify whether the arc is filled:

- **yes**: The arc is filled with the specified drawing color.
- **no**: Only the outline of the arc is drawn (default).

### Usage

The resulting arc begins at `startAngle` and extends for `arcAngle` degrees. Degrees start at 0 in the three o'clock position. A positive value indicates a counter-clockwise rotation; a negative value indicates a clockwise rotation. The center of the arc is the center of the rectangle whose origin is \((x,y)\) and whose size is specified by the `width` and `height` parameters. The angles are specified relative to the non-square extents of the bounding rectangle so that 45 degrees always falls on the line from the center of the ellipse to the upper-right corner of the bounding rectangle. As a result, if the bounding rectangle is noticeably longer on one axis than the other, the angles to the start and end of the arc segment are skewed farther along the longer axis of the bounds. If the `filled` parameter is set to **yes**, the area inside the oval is filled with the current drawing color. Use the `ImageSetDrawingColor` and `ImageSetDrawingStroke` functions to specify the color and line attributes of the arc. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

### Example

```coldfusion
<!--- This example shows how to use the ImageNew function to create a blank ColdFusion image that is 250 pixels wide and 180 pixels high. --->
cfset myImage=ImageNew("",250,320)

<!--- Set the drawing color for the arc to orange. --->
cfset ImageSetDrawingColor(myImage,"orange")

<!--- Turn on antialiasing to improve image quality. --->
cfset ImageSetAntialiasing(myImage,"on")

<!--- Draw an enclosed orange arc starting at the coordinate (5,5). --->
cfset ImageDrawArc(myImage,5,5,200,300,100,100,"yes")

<!--- Display the image in a browser. --->
cfimage action="writeToBrowser" source="#myImage#"
```
ImageDrawBeveledRect

Description

Draws a rectangle with beveled edges.

Returns

Nothing.

Category

Image functions

Function syntax

ImageDrawBeveledRect(name, x, y, width, height, raised [, filled])

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate of the rectangle.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate of the rectangle.</td>
</tr>
<tr>
<td>width</td>
<td>Required. The width of the rectangle.</td>
</tr>
<tr>
<td>height</td>
<td>Required. The height of the rectangle.</td>
</tr>
</tbody>
</table>
| raised    | Required. Specify whether the rectangle appears raised above the surface or sunk into the surface:  
* yes: The rectangle is raised.  
* no: The rectangle is sunk (default). |
filled

<table>
<thead>
<tr>
<th>Optional. Specify whether the rectangle is filled:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• yes: The rectangle is filled with the specified drawing color.</td>
</tr>
<tr>
<td>• no: Only the outline of the rectangle is drawn (default).</td>
</tr>
</tbody>
</table>

Usage

The edges of the rectangle are highlighted so that they appear beveled and lit from the upper-left corner. The colors used for the highlighting effect are determined by the current drawing color. If the filled parameter is set to yes, the cuboid area is filled with the current drawing color. Use the `ImageSetDrawingColor` and `ImageSetDrawingStroke` functions to specify the color and line attributes of the rectangle. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example

```coldfusion
<!--- This example shows how to create a bevel-edged rectangle. --->
<!--- Use the ImageNew function to create a 200x200-pixel image. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the drawing color for the image to light gray. --->
<cfset ImageSetDrawingColor(myImage,"lightgray")>
<!--- Draw a 3D gray, filled, raised rectangle. --->
<cfset ImageDrawBeveledRect(myImage,50,50,100,75,"yes","yes")>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
ImageDrawCubicCurve

Description

Draws a cubic curve.

Returns

Nothing.

Category

Image functions

Function syntax

ImageDrawCubicCurve(name, ctrlx1, ctrly1, ctrlx2, ctrly2, x1, y1, x2, y2)

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>ctrlx1</td>
<td>Required. The x coordinate of the first control point of the cubic curve segment.</td>
</tr>
<tr>
<td>ctrly1</td>
<td>Required. The y coordinate of the first control point of the cubic curve segment.</td>
</tr>
<tr>
<td>ctrlx2</td>
<td>Required. The x coordinate of the second control point of the cubic curve segment.</td>
</tr>
<tr>
<td>ctrly2</td>
<td>Required. The y coordinate of the second control point of the cubic curve segment.</td>
</tr>
<tr>
<td>x1</td>
<td>Required. The x coordinate of the start point of the cubic curve segment.</td>
</tr>
<tr>
<td>y1</td>
<td>Required. The y coordinate of the start point of the cubic curve segment.</td>
</tr>
</tbody>
</table>
Adobe ColdFusion Documentation

x2

Required. The x coordinate of the end point of the cubic
curve segment.

y2

Required. The y coordinate of the end point of the cubic
curve segment.

Usage

Coordinates can be integers or real numbers.

Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to specify the color and line
attributes of the cubic curve. Use the ImageSetAntialiasing function to improve the quality of the rendered
image.
Example

<!--- This example shows how to draw a curved line that looks like a stylized 7.
--->
<!--- Use the ImageNew function to create a blank ColdFusion image that is 200
pixels wide and 380 pixels high. --->
<cfset myImage=ImageNew("",200,380)>
<!--- Set the drawing color to magenta. --->
<cfset ImageSetDrawingColor(myImage,"magenta")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Draw a curved line that starts at (380,28) and ends at (32,56) with its first
control point at (120,380) and its second control point at (5,15). --->
<cfset ImageDrawCubicCurve(myImage,120,380,5,15,380,28,32,56)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">

© 2014 Adobe Systems Incorporated. All rights reserved.

1809


**ImageDrawLine**

**Description**

Draws a single line defined by two sets of x and y coordinates on a ColdFusion image.

**Returns**

Nothing.

**Category**

Image functions

**Function syntax**

```
ImageDrawLine(name, x1, y1, x2, y2)
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x1</td>
<td>Required. The x coordinate for the start point of the line.</td>
</tr>
<tr>
<td>y1</td>
<td>Required. The y coordinate for the start point of a line.</td>
</tr>
<tr>
<td>x2</td>
<td>Required. The x coordinate for the end point of the line.</td>
</tr>
<tr>
<td>y2</td>
<td>Required. The y coordinate for the end point of the line.</td>
</tr>
</tbody>
</table>

**Usage**

Each pair of coordinates, (x1,y1) and (x2,y2), defines a point. The start and end points cannot be the same. This function is affected by the attributes defined in the ImageSetDrawingStroke and ImageSetDrawingColor functions. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

**Example**
<!--- This example shows how to draw a square bisected by a diagonal line. --->
<!--- Use the ImageNew function to create a 100x100-pixel image. --->
<cfset myImage=ImageNew("",100,100)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Draw a diagonal line that bisects the square. --->
<cfset ImageDrawLine(myImage,0,0,100,100)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
**ImageDrawLines**

**Description**

Draws a sequence of connected lines defined by arrays of x and y coordinates.

**Returns**

Nothing.

**Category**

Image functions

**Function syntax**

```
ImageDrawLines(name, xcoords, ycoords [, isPolygon, filled])
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>xcoords</td>
<td>Required. A CFML array of x coordinates.</td>
</tr>
<tr>
<td>ycoords</td>
<td>Required. A CFML array of y coordinates.</td>
</tr>
<tr>
<td>isPolygon</td>
<td>Optional. Specify whether the lines form a polygon:</td>
</tr>
<tr>
<td></td>
<td>• yes: The lines are connected to form a polygon.</td>
</tr>
<tr>
<td></td>
<td>• no: The lines do not form a polygon (default).</td>
</tr>
<tr>
<td>filled</td>
<td>Optional. Specify whether the polygon is filled:</td>
</tr>
<tr>
<td></td>
<td>• yes: The polygon is filled with the specified drawing color.</td>
</tr>
<tr>
<td></td>
<td>• no: Only the outline of the polygon is drawn (default).</td>
</tr>
</tbody>
</table>

**Usage**

Each pair of (x,y) coordinates defines a point. To draw a polygon, set `isPolygon` to `yes`. The start point cannot be
the same value as the end point. If `isPolygon` is `yes`, a line joining start point and the end point is drawn to complete a polygon. If `isPolygon` is `no`, line completing the polygon is not drawn. Set the `isPolygon` and `filled` parameters to `yes` to draw a polygon filled with the current drawing color. Use the `ImageSetDrawingColor` and `ImageSetDrawingStroke` functions to control the color and line attributes. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example

```coldfusion
<!--- This example shows how to draw a zigzag line. --->
<!--- Use the ImageNew function to create a 250x250-pixel image. --->
<cfset myImage=ImageNew("",250,250)>
<!--- Set the drawing color to cyan. --->
<cfset ImageSetDrawingColor(myImage,"cyan")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Create arrays for the x and y coordinates. --->
<cfset x = ArrayNew(1)>
<cfset y = ArrayNew(1)>
<!--- Set the values for the x and y coordinates for three connected line segments: the first segment begins at (100,50) and ends at (50,100). The second segment begins at (50, 100) and ends at (200,100). The third segment begins at (200,100) and ends at (100,200). --->
<cfset x[1] = "100">
<cfset x[2] = "50">
<cfset x[3] = "200">
<cfset x[4] = "100">
<cfset y[1] = "50">
<cfset y[2] = "100">
<cfset y[3] = "100">
<cfset y[4] = "200">
<!--- Draw the lines on the image. --->
<cfset ImageDrawLines(myImage,x,y)>
<!--- Display the image in a browser. --->
<cfimage source=#myImage# action="writeToBrowser">
ImageDrawOval

Description

Draws an oval.

Returns

Nothing.

Category

Image functions

Function syntax

```
ImageDrawOval(name, x, y, width, height [, filled])
```

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate of the upper left corner of the oval to draw.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate of the upper left corner of the oval to draw.</td>
</tr>
<tr>
<td>width</td>
<td>Required. The width of the oval to draw.</td>
</tr>
<tr>
<td>height</td>
<td>Required. The height of the oval to draw.</td>
</tr>
</tbody>
</table>
| filled    | Optional. Specify whether the oval is filled:  
  - yes: The oval is filled with the specified drawing color.  
  - no: Only the outline of the oval is drawn (default). |

Usage
The result is a circle or ellipse that fits within the rectangle specified by the x, y, width, and height arguments. If the filled parameter is set to yes, the area inside the oval is filled with the current drawing color. Use the `ImageSetDrawingColor` and `ImageSetDrawingStroke` functions to specify the color and line attributes of the oval. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example

Example 1

```cfml
<!--- This example shows how to draw a green filled oval. --->
<!--- Use the ImageNew function to create a 200x110-pixel image. --->
<cfset myImage=ImageNew("",200,110)>
<!--- Set the drawing color to green. --->
<cfset ImageSetDrawingColor(myImage,"green")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Draw a filled green oval on the image. --->
<cfset ImageDrawOval(myImage,5,5,190,100,"yes")>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```cfml
<!--- This example shows how to draw a red circle with a line through it. --->
<!--- Use the ImageNew function to create a 201x201-pixel image. --->
<cfset myImage=ImageNew("",201,201)>
<!--- Set the drawing color to red. --->
<cfset ImageSetDrawingColor(myImage,"red")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the line width to 10 pixels. --->
<cfset attr=StructNew()>
<cfset attr.width = 10>
<cfset ImageSetDrawingStroke(myImage,attr)>
<!--- Draw a diagonal line starting at (40,40) and ending at (165,165) on myImage. --->
<cfset ImageDrawLine(myImage,40,40,165,165)>
<!--- Draw a circle starting at (5,5) and is 190 pixels high and 190 pixels wide. --->
<cfset ImageDrawOval(myImage,5,5,190,190)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```
ImageDrawPoint

Description

Draws a point at the specified (x,y) coordinate.

Returns

Nothing.

Category

Image functions

Function syntax

ImageDrawPoint(name, x, y)

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate of the point.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate of the point.</td>
</tr>
</tbody>
</table>

Usage

Use the ImageSetDrawingStroke and ImageSetDrawingColor functions to control the appearance of the drawing point. For example, set the width attribute of the ImageSetDrawingStroke function to 10 pixels to draw a 20-pixel-square centered at (x,y). Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
<!--- This example shows how to draw a large square in the middle of an image. --->
<!--- Use the ImageNew function to create a 200x200-pixel image. --->
<cfset myImage=ImageNew('"",200,200)>
<!--- Set the drawing color to orange. --->
<cfset ImageSetDrawingColor(myImage,"orange")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the drawing area to 10 pixels. --->
<cfset attr = StructNew()>
<cfset attr.width = 10>
<cfset ImageSetDrawingStroke(myImage,attr)>
<!--- Draw the point at (100,100). --->
<cfset ImageDrawPoint(myImage,100,100)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
ImageDrawQuadraticCurve

Description

Draws a curved line. The curve is controlled by a single point.

Returns

Nothing.

Category

Image functions

Function syntax

ImageDrawQuadraticCurve(name, ctrlx1, ctrly1, ctrlx2, ctrly2, x1, y1, x2, y2)

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>ctrlx1</td>
<td>Required. The x coordinate of the first control point of the quadratic curve segment.</td>
</tr>
<tr>
<td>ctrly1</td>
<td>Required. The y coordinate of the first control point of the quadratic curve segment.</td>
</tr>
<tr>
<td>ctrlx2</td>
<td>Required. The x coordinate of the second control point of the quadratic curve segment.</td>
</tr>
<tr>
<td>ctrly2</td>
<td>Required. The y coordinate of the second control point of the quadratic curve segment.</td>
</tr>
<tr>
<td>x1</td>
<td>Required. The x coordinate of the start point of the quadratic curve segment.</td>
</tr>
<tr>
<td>y1</td>
<td>Required. The y coordinate of the start point of the quadratic curve segment.</td>
</tr>
<tr>
<td>x2</td>
<td>Required. The x coordinate of the end point of the quadratic curve segment.</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>y2</td>
<td>Required. The y coordinate of the end point of the quadratic curve segment.</td>
</tr>
</tbody>
</table>

**Usage**

A quadratic curve is a curve controlled by a single control point. The curve is drawn from the last point in the shape to the target x and y coordinates. Coordinates can be integers or real numbers. Use the `ImageSetDrawingColor` and `ImageSetDrawingStroke` functions to specify the color and lines of the quadratic curve. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

**Example**

```cfml
<!--- This example shows how to draw a curved line. --->
<!--- Use the ImageNew function to create a 400x400-pixel image. --->
<cfset myImage=ImageNew("",400,400)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the drawing color to green. --->
<cfset ImageSetDrawingColor(myImage,"green")>
<!--- Draw a curved line on the image. --->
<cfset ImageDrawQuadraticCurve(myImage,120,320,5,15,380,280)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```
**ImageDrawRect**

**Description**

Draws a rectangle.

**Returns**

Nothing.

**Category**

Image functions

**Function syntax**

```
ImageDrawRect(name, x, y, width, height [, filled])
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate of the rectangle.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate of the rectangle.</td>
</tr>
<tr>
<td>width</td>
<td>Required. The width of the rectangle.</td>
</tr>
<tr>
<td>height</td>
<td>Required. The height of the rectangle.</td>
</tr>
</tbody>
</table>
| filled    | Optional. Specify whether the rectangle is filled:  
- yes: The rectangle is filled with the specified drawing color.  
- no: Only the outline of the rectangle is drawn (default). |

**Usage**
The left and right edges of the rectangle are at x and x plus width, respectively. The upper and lower edges are at y and y plus height, respectively. Set the filled parameter to yes to fill the rectangle with the current drawing color. Use the ImageSetDrawingColor and ImageSetDrawingStroke functions to format the color and lines of the rectangle. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example

```cfml
<!--- This example shows how to draw a rectangle. --->
<!--- Use the ImageNew function to create a ColdFusion image that is 150 pixels wide and 200 pixels high. --->
<cfset myImage=ImageNew("",150,200)>
<!--- Set the drawing color for the image to yellow. --->
<cfset ImageSetDrawingColor(myImage,"yellow")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Draw a filled yellow rectangle on the image. --->
<cfset ImageDrawRect(myImage,25,45,100,20,"yes")>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
**ImageDrawRoundRect**

**Description**

Draws a rectangle with rounded corners.

**Returns**

Nothing.

**Category**

Image functions

**Function syntax**

```plaintext
ImageDrawRoundRect(name, x, y, width, height, arcWidth, arcHeight [, filled])
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate of the rectangle.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate of the rectangle.</td>
</tr>
<tr>
<td>width</td>
<td>Required. The width of the rectangle.</td>
</tr>
<tr>
<td>height</td>
<td>Required. The height of the rectangle.</td>
</tr>
<tr>
<td>arcWidth</td>
<td>Required. The horizontal diameter of the arc at the four corners.</td>
</tr>
<tr>
<td>arcHeight</td>
<td>Required. The vertical diameter of the arc at the four corners.</td>
</tr>
<tr>
<td>filled</td>
<td>Optional. Specify whether the rectangle is filled:</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>• yes: The rectangle is filled with the specified drawing color.</td>
</tr>
<tr>
<td></td>
<td>• no: Only the outline of the rectangle is drawn (default).</td>
</tr>
</tbody>
</table>

Usage

The left and right edges of the rectangle are at x and x plus width, respectively. The upper and lower edges are at y and y plus height, respectively. Set the `filled` parameter to `yes` to fill the rectangle with the current drawing color. Use the `ImageSetDrawingColor` and `ImageSetDrawingStroke` functions to control the color and line attributes of the rectangle. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example

Example 1

```html
<!---- This example shows how to draw a square with rounded corners. ---->
<!---- Create a 200x200-pixel image. ---->
<cfset myImage=ImageNew("",200,200)>
<!---- Set the drawing color for the image to blue. ---->
<cfset ImageSetDrawingColor(myImage,"blue")>
<!---- Turn on antialiasing to improve image quality. ---->
<cfset ImageSetAntialiasing(myImage,"on")>
<!---- Draw a blue filled square with round corners of arcWidth=10 and arcHeight=2. ---->
<cfset ImageDrawRoundRect(myImage,5,5,190,190,"yes",10,2)>
<!---- Display the image in a browser. ---->
<cfimage source="#myImage#" action="writeToBrowser"/>
```

Example 2
<cfset myImage = imageNew("",100,100,"argb")>
<cfset textStruct = structNew()>
<cfset textStruct.size=16>
<cfset textStruct.style="bold">
<cfset textStruct.font="Trebuchet MS">

<cfoutput>
<cfloop from="1" to="20" index="i">
  <!--- Turn on antialiasing to improve the quality of the rendered image. --->
  <cfset ImageSetAntialiasing(myImage,"on")>
  <!--- Set the background color. --->
  <cfset ImageSetBackgroundColor(myImage,"cyan") />
  <!--- Set the drawing color. --->
  <cfset ImageSetDrawingColor(myImage,"black") />
  <!--- Draw a rectangle with rounded corners. --->
  <cfset ImageDrawRoundRect(myImage,10,10,myImage.width-20,myImage.height-20,i,i,"yes")>
  <!--- Set the text arc value. --->
  <cfset ImageSetDrawingColor(myImage,"##cccccc")>
  <cfset ImageDrawText(myImage, "#i#",30,30,textStruct)>
  <!--- Write the image to a file. --->
  <cfset imageWrite(myImage,"##expandPath("##i#.png")##")>
  <!--- Display the image. --->
  <img src="##i#.png">
</cfloop>
</cfoutput>
**ImageDrawText**

**Description**

Draws a text string on a ColdFusion image with the baseline of the first character positioned at (x,y) in the image.

**Returns**

A struct that contains width and height of the text drawn.

**Category**

*Image functions*

**Function syntax**

```
ImageDrawText(name, str, x, y [, attributeCollection])
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>str</td>
<td>Required. The text string to draw.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate for the start point of the string.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate for the start point of the string.</td>
</tr>
<tr>
<td>attributeCollection</td>
<td>Optional. The structure used to specify the text characteristics. See the Usage section.</td>
</tr>
</tbody>
</table>

**Usage**

Specify all the optional key-value pairs in an attributeCollection structure. To specify the text color, use the ImageSetDrawingColor function.

attributeCollection
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>font</td>
<td>The name of the font used to draw the text string. If you do not specify the font attribute, the text is drawn in the default system font.</td>
</tr>
<tr>
<td>size</td>
<td>The font size for the text string. The default value is 10 points.</td>
</tr>
<tr>
<td>style</td>
<td>The style to apply to the font:</td>
</tr>
<tr>
<td></td>
<td>• bold</td>
</tr>
<tr>
<td></td>
<td>• italic</td>
</tr>
<tr>
<td></td>
<td>• boldItalic</td>
</tr>
<tr>
<td></td>
<td>• plain (default)</td>
</tr>
<tr>
<td>strikethrough</td>
<td>Specify whether strikethrough is applied to the text image:</td>
</tr>
<tr>
<td></td>
<td>• yes: For vertical text, strikethrough is applied to each character individually.</td>
</tr>
<tr>
<td></td>
<td>• no (default)</td>
</tr>
<tr>
<td>underline</td>
<td>Specify whether underline is applied to the text image:</td>
</tr>
<tr>
<td></td>
<td>• yes: For vertical text, underline is applied to each character individually.</td>
</tr>
<tr>
<td></td>
<td>• no (default)</td>
</tr>
</tbody>
</table>

Example

Example 1

```html
<!--- This example shows how to create a text string image. --->
<!--- Use the ImageNew function to create a 200x100-pixel image. --->
<cfset myImage=ImageNew("",200,100)>
<!--- Set the drawing color to green. --->
<cfset ImageSetDrawingColor(myImage,"green")>
<!--- Specify the text string and the start point for the text. --->
<cfset ImageDrawText(myImage,"It's not easy being green.",10,50)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2
<!--- This example shows how to draw three text strings with different text attributes. --->
<!--- Use the ImageNew function to create a 400x400-pixel image. --->
<cfset myImage=ImageNew("",400,400)>
<!--- Set the text attributes. --->
<cfset attr = StructNew()>
<cfset attr.underline = "yes">
<cfset attr.size = 25>
<cfset attr.style = "bold">
<cfset ImageSetDrawingColor(myImage,"yellow")>
<!--- Draw the text string "ColdFusion Rocks!" starting at (100,150). --->
<cfset ImageDrawText(myImage,"ColdFusion Rocks!",100,150,attr)>
<!--- Set new text attributes. --->
<cfset attr=StructNew()>
<cfset attr.size = 18>
<cfset attr.strikethrough = "yes">
<cfset attr.style = "bolditalic">
<cfset ImageSetDrawingColor(myImage,"red")>
<!--- Draw the text string "Powered by ColdFusion" starting at (100,200). --->
<cfset ImageDrawText(myImage,"Powered by ColdFusion",110,200,attr)>
<!--- Set new text attributes. --->
<cfset attr = StructNew()>
<cfset attr.font="Arial">
<cfset attr.style="italic">
<cfset attr.size=15>
<cfset ImageSetDrawingColor(myImage,"white")>
<!--- Draw the text string "Coming in 2007" starting at (150,250). --->
<cfset ImageDrawText(myImage,"We've arrived",150,250,attr)>
<!--- Display the text image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser"
**ImageFlip**

**Description**

Flips an image across an axis.

**Returns**

Nothing.

**Category**

Image functions

**Function syntax**

```
ImageFlip(name [, transpose])
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>transpose</td>
<td>Optional. Transpose the image:</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>vertical</td>
<td>Flip an image across an imaginary horizontal line that runs through the center of the image (default).</td>
</tr>
<tr>
<td>horizontal</td>
<td>Flip an image across an imaginary vertical line that runs through the center of the image.</td>
</tr>
<tr>
<td>diagonal</td>
<td>Flip an image across its main diagonal that runs from the upper-left to the lower-right corner.</td>
</tr>
<tr>
<td>antidiagonal</td>
<td>Flip an image across its main diagonal that runs from the upper-right to the lower-left corner.</td>
</tr>
</tbody>
</table>

- **transpose**: 180
- **270**: Rotate an image clockwise by 90, 180, or 270 degrees.

**Usage**

If you do not specify the `transpose` parameter for the `ImageFlip` function, the image is transposed on a vertical axis, creating an image that is an upside-down version of the source. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

**Example**

**Example 1**

```cfml
<!--- This example shows how to rotate an image by 270 degrees. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Rotate the image by 270 degrees. --->
<cfset ImageFlip(myImage,"270")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">```

**Example 2**
Example 3

<!--- This example shows how to flip an image on a vertical axis. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Flip the image so that it is upside down. --->
<cfset ImageFlip(myImage,"vertical")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">

Example 4

<!--- This example shows how to flip an image on a horizontal axis. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Flip the image so that it is a mirror image of the source. --->
<cfset ImageFlip(myImage,"horizontal")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">

Example 5

<!--- This example shows how to flip an image on a diagonal axis. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Flip the image on a diagonal axis. --->
<cfset ImageFlip(myImage,"diagonal")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
**ImageGetBlob**

**Description**

Retrieves the bytes of the underlying image. The bytes are in the same image format as the source image.

**Returns**

The bytes of the underlying image of a BLOB.

**Category**

*Image functions*

**Function syntax**

```
ImageGetBlob(source)
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to insert ColdFusion images into BLOB columns of databases.

⚠️ **Note**

If you do not specify a source image, you get a parameter validation error.

**Example**

Example 1
<!--- This example shows how to add a ColdFusion image to a database. --->
<!--- Create ColdFusion image from a an existing JPEG file. --->
<cfimage source="aiden01.jpg" name="myImage">
<!--- Use the cfquery tag to insert the ColdFusion image as a BLOB in the database. --->
<cfquery name="InsertBlobImage" datasource="myBlobData">
    INSERT INTO EMPLOYEES (FirstName, LastName, Photo)
    VALUES ('Aiden', 'Quinn', #ImageGetBlob(myImage)#)
</cfquery>

Example 2The following example shows how to use the ImageNew function to generate thumbnail images in JPEG format from BLOB data retrieved from a database:

<!--- Use the cfquery tag to retrieve all employee photos and employee IDs from a database. --->
<cfquery name="GetBLOBs" datasource="myBlobData">
    SELECT EMPLOYEEID, PHOTO FROM Employees
</cfquery>
<!--- Use the ImageNew function to create a ColdFusion image from the BLOB data that was retrieved from the database. --->
<cfset myImage = ImageNew(#GetBLOBs.PHOTO#)>
<!--- Create thumbnail versions of the images by resizing them to a 100x100-pixel image, while maintaining the aspect ratio of the source image. --->
<cfset ImageScaleToFit(myImage,100,"")>
<!--- Convert the images to JPEG format and save them to files in the thumbnails subdirectory, using the employee ID as the filename. --->
<cfimage source="#myImage#" action="write"
        destination="images/thumbnails/#GetBLOBs.EMPLOYEEID#.jpg">
**ImageGetBufferedImage**

**Description**

Returns the `java.awt.BufferedImage` object underlying the current ColdFusion image.

**Returns**

The `java.awt.BufferedImage` object.

**Category**

Image functions

**Function syntax**

```
ImageGetBufferedImage(name)
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to return an image object that can be used with other Java Abstract Windowing Toolkit (AWT) objects embedded in the page.

**Example**
<!--- This example shows how to create a ColdFusion image, modify it, and retrieve the width of the buffered image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul05.jpg" name="myImage">
<!--- Blur the image by an order of 10. --->
<cfset ImageBlur(myImage,10)>
<!--- Get the blurred image from the buffer and set it to variable x. --->
<cfset x = ImageGetBufferedImage(myImage)>
<!--- Return the width of the buffered image. --->
<cfoutput>#x.getWidth()#</cfoutput>
ImageGetEXIFMetadata

Description

Retrieves the Exchangeable Image File Format (EXIF) headers in an image as a CFML structure.

Returns

A structure with the EXIF header values.

Category

Image functions

Function syntax

```
ImageGetEXIFMetadata(name)
```

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>

Usage

The EXIF is a standard for storing interchange information in image files, especially those using JPEG compression. Most digital cameras use the EXIF format. EXIF metadata includes information pertaining to the creation of the image, such as the creation date, the software used to create the image, the aperture, the make and model, and the resolution of the image. The result of the ImageGetEXIFMetadata function is cached in the ColdFusion image to optimize performance. The ImageGetEXIFMetadata function applies only to JPEG images. If you try to retrieve metadata for Base64, BLOB, or other types of images, ColdFusion generates errors.

Example
<!--- This example shows how to retrieve the EXIF header information from a JPEG file. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="images\paul05.jpg" name="myImage">
<!--- Retrieve the metadata associated with the image. --->
<cfset data = ImageGetEXIFMetadata(myImage)>
<!--- Display the ColdFusion image parameters. --->
<cfdump var="#myImage#">
<!--- Display the EXIF header information associated with the image (creation date, software, and so on). --->
<cfdump var="#data#"/>
**ImageGetEXIFTag**

**Description**

Retrieves the specified EXIF tag in an image.

**Returns**

The value of the specified EXIF tag.

**Category**

*Image functions*

**Function syntax**

```plaintext
ImageGetEXIFTag(name, tagName)
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>tagName</td>
<td>Required. The EXIF tag name to be returned.</td>
</tr>
</tbody>
</table>

**Usage**

The `ImageGetEXIFTag` function applies only to JPEG images. If you try to retrieve metadata for Base64, BLOB, or other types of images, ColdFusion generates errors.

**Example**
<!--- This example shows how to retrieve one element from the EXIF information associated with an image. --->

<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul05.jpg" name="myImage">

<!--- Retrieve the name of the software application used to create the original image. --->
<cfset data = ImageGetEXIFTag(myImage,"software")>

<!--- Display the name of the software. --->
<cfdump var="#data#">
**ImageGetHeight**

**Description**

Retrieves the height of the ColdFusion image in pixels.

**Returns**

The height of the specified ColdFusion image in pixels.

**Category**

Image functions

**Function syntax**

```
ImageGetHeight(name)
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to retrieve the height of a ColdFusion image.

**Example**

```html
<!---- This example shows how to retrieve the height of an image. --->
<!---- Create a ColdFusion image from a JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!---- Retrieve the height of the image. --->
<cfset height=ImageGetHeight(myImage)>
<!---- Display the height of the image. --->
<cfdump var="#height#"/>
```
ImageGetIPTCMetadata

Description

Retrieves the International Press Telecommunications Council (IPTC) headers in a ColdFusion image as a structure. The IPTC metadata contains text that describes the image that is stored with it. IPTC metadata includes, but is not limited to, caption, keywords, credit, copyright, object name, created date, byline, headline, and source.

Returns

A structure containing IPTC header values.

Category

Image functions

Function syntax

ImageGetIPTCMetadata(name)

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>

Usage

The IPTC metadata contains text that describes the image that is stored with it. IPTC metadata includes, but is not limited to, caption, keywords, credit, copyright, object name, created date, byline, headline, and source. The result of the ImageGetIPTCMetadata function is cached in the ColdFusion image to optimize performance. The ImageGetIPTCMetadata function applies only to JPEG images. If you try to retrieve metadata for Base64, BLOB, or other types of images, ColdFusion generates errors.

Example
<!--- This example shows how to retrieve the IPTC header information for a JPEG file. --->

<!--- Create a ColdFusion image from a JPEG file. --->
<cfimage source="images\aiden01.jpg" name="myImage">

<!--- Retrieve the IPTC header information saved with the image, such as copyright, caption, and headline. --->
<cfset data = ImageGetIPTCMetadata(myImage)>

<!--- Display the parameter values for the ColdFusion image. --->
<cfdump var="#myImage#">

<!--- Display the IPTC header information. --->
<cfdump var="#data#"
ImageGetIPTCTag

Description

Retrieves the value of the IPTC tag for a ColdFusion image.

Returns

The value of the IPTC tag.

Category

Image functions

Function syntax

```
ImageGetIPTCTag(name, tagName)
```

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>tagName</td>
<td>Required. The IPTC tag name whose value is returned.</td>
</tr>
</tbody>
</table>

Usage

The `ImageGetIPTCTag` function applies only to JPEG images. If you try to retrieve metadata for Base64, BLOB, or other image types, ColdFusion generates errors.

Example

```
<!--- This example shows how to retrieve the caption for a JPEG file. --->
<!--- Create a ColdFusion image from a JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul05.jpg" name="myImage" action="read">
<!--- Retrieve the camera make used to take the original picture. --->
<cfset cameraMake=ImageGetIPTCTag(myImage,"make")>
<cfdump var="#cameraMake#">
```
ImageGetWidth

Description

Retrieves the width of the specified ColdFusion image.

Returns

An integer that represents the width of the ColdFusion image in pixels.

Category

Image functions

Function syntax

```
ImageGetWidth(name)
```

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>

Example

```
<!--- This example shows how to retrieve the width of an image. --->
<!--- Create a ColdFusion image from an existing JPEG file.--->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Get the width of the image. --->
<cfset width=#ImageGetWidth(myImage)#>
<!--- Display the width of the image in pixels. --->
<cfdump var=#width#>
```
ImageGrayscale

Description

Converts a ColdFusion image to grayscale.

Returns

Nothing.

Category

Image functions

Function syntax

ImageGrayscale(name)

See also

ImageBlur, ImageNegative, ImageSetAntialiasing, ImageSharpen, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>

Usage

Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example

<!---- This example shows how to change a color image to grayscale. --->
<!---- Create a ColdFusion image from an existing color image. --->
<cfimage source="./cfdocs/images/artgallery/jeff04.jpg" name="myImage">
<!---- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!---- Change the image to grayscale. --->
<cfset ImageGrayscale(myImage)>
<!---- Save the grayscale image to a JPEG file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpg" overwrite="yes">
<!---- Display the source image and the grayscale image. --->
<img src="../cfdocs/images/artgallery/jeff04.jpg"/>
<img src="test_myImage.jpg"/>
ImageInfo

Description

Returns a structure that contains information about the image, such as height, width, color model, size, and filename.

Returns

A structure that contains information for image parameters.

Category

Image functions

Function syntax

ImageInfo(name)

See also

cfimage, ImageGetHeight, ImageGetWidth, IsImage, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>

Usage

Use this function to determine whether images are compatible. For example, to use the ImageOverlay function to overlay two images, both images must have the same color model.

Example
<!--- This example shows how to retrieve information associated with the image. --->

<!--- Create a ColdFusion image from a JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Retrieve the information associated with the image. --->
<cfset info=ImageInfo(myImage)>
<cfdump var="#info#"></cfdump>
<p>height = <cfoutput>#{info.height#}</cfoutput>
<p>width = <cfoutput>#{info.width#}</cfoutput>
<p>source = <cfoutput>#{info.source#}</cfoutput>
<p>pixel size = <cfoutput>#{info.colormodel.pixel_size#}</cfoutput>
<p>transparency = <cfoutput>#{info.colormodel.transparency#}</cfoutput>
ImageMakeColorTransparent

Description

Creates an image and sets a transparent color.

Returns

Image object

Syntax

imageMakeColorTransparent(img, color__)

History

ColdFusion 10: Added this function

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>img</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>
| color     | Required. The transparent color:  
- Hexadecimal value of RGB color. For example, specify the color white as "##FFFFFF" or "FFFFFF".  
- String value of color (for example, "black", "red", "green").  
- The default value of the transparent color is "black". List of three numbers for (R,G,B) values. Each value must be in the range 0-255. |

Example

```cfc
cfset myImage=ImageNew("",200,110)
<!--- Set the drawing color to green. --->
cfset ImageSetDrawingColor(myImage,"green")
<!--- Turn on antialiasing to improve image quality. --->
cfset ImageSetAntialiasing(myImage,"on")
<!--- Draw a filled green oval on the image. --->
cfset ImageDrawOval(myImage,5,5,190,100,"yes")
<!--- Display the image in a browser. --->
cfoutput<PNG image<br></cfoutput>
cfset ImageWrite(myImage,"#expandpath('.')#/png.png")
cfset myImage = ImageRead("#expandpath('.')#/png.png")
cfimage source="#myImage#" action="writeToBrowser" >
cfset x =ImageMakeColorTransparent(myImage,"green")
cfimage source="#x#" action="writeToBrowser" >
```
ImageMakeTranslucent

Description

Create a new translucent image with given percentage of translucence.

Returns

Image object

History

ColdFusion 10: Added this function

Syntax

imageMakeTranslucent (img, percent)

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>img</td>
<td>Required. Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>percentage</td>
<td>Required. The percent of translucence:</td>
</tr>
<tr>
<td></td>
<td>• 0 = opaque</td>
</tr>
<tr>
<td></td>
<td>• 100 = transparent</td>
</tr>
<tr>
<td></td>
<td>Decimal values are supported.</td>
</tr>
</tbody>
</table>

Example

The following example illustrates three images with the second one translucent than first and the third one translucent than the second.
<cfset myImage=ImageNew("",200,110)>

<!--- Set the drawing color to green. --->
<cfset ImageSetDrawingColor(myImage,"green")>

<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>

<!--- Draw a filled green oval on the image. --->
<cfset ImageDrawOval(myImage,5,5,190,100,"yes")>

<!--- Display the image in a browser. --->
<cfoutput>PNG image<br></cfoutput>
<cfset ImageWrite(myImage,"#expandpath('.')#/png.png")>
<cfset myImage = ImageRead("#expandpath('.')#/png.png")>
<cfimage source="#myImage#" action="writeToBrowser" >
<cfset x =ImageMakeTranslucent(myImage,50)>
<cfimage source="#x#" action="writeToBrowser" >
<cfset x =ImageMakeTranslucent(myImage,75)>
<cfimage source="#x#" action="writeToBrowser" >
<cfset x =ImageMakeTranslucent(myImage,100)>
<cfimage source="#x#" action="writeToBrowser" >
</body></html?
ImageNegative

Description

Inverts the pixel values of a ColdFusion image.

Returns

Nothing.

Category

Image functions

Function syntax

ImageNegative(name)

See also

ImageBlur, ImageGrayscale, ImageSetAntialiasing, ImageSharpen, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>

Usage

The resulting image has the same dimensions of the source image, but not necessarily the same number bytes. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
<!--- This example shows how to create a negative version of an image. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Create a negative version of the image. --->
<cfset ImageNegative(myImage)>
<!--- Save the modified image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpg" overwrite="yes">
<!--- Display the source image and the negative image. --->
<img src="../cfdocs/images/artgallery/jeff05.jpg"/>
<img src="test_myImage.jpg"/>
**ImageNew**

**Description**

Creates a ColdFusion image.

**Returns**

A ColdFusion image.

**Category**

*Image functions*

**Function syntax**

```
ImageNew([source, width, height, imageType, canvasColor])
```

**See also**

*cfimage, ImageCopy, ImageRead, ImageReadBase64, ImageSetDrawingColor, IsImageFile*

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Optional. The source image on-disk or in-memory pathname, URL, a ColdFusion variable that is read into the new ColdFusion image, or a Java buffered image.</td>
</tr>
<tr>
<td>width</td>
<td>Optional. The width of the new ColdFusion image. Valid when the height is specified and the source is not.</td>
</tr>
<tr>
<td>height</td>
<td>Optional. The height of the new ColdFusion image. Valid when the width is specified and the source is not.</td>
</tr>
<tr>
<td>imageType</td>
<td>Optional. The type of the ColdFusion image to create:</td>
</tr>
<tr>
<td></td>
<td>• rgb</td>
</tr>
<tr>
<td></td>
<td>• argb</td>
</tr>
<tr>
<td></td>
<td>• grayscale</td>
</tr>
<tr>
<td></td>
<td>Valid only when width and height are specified.</td>
</tr>
</tbody>
</table>
canvasColor

Optional. Color of the image canvas:

- Hexadecimal value of RGB color. For example, specify the color white as `##FFFF00` or `FFFF00`.
- String value of color (for example, "black", "red", "green"). The default value of the drawing color is "black".
- List of three numbers for (R,G,B) values. Each value must be in the range 0-255.

Usage

You can pass the `ImageNew` function any of the following parameters:

- Absolute or relative pathname: The image file located at the specified pathname on a disk is read and returned as a ColdFusion image.
- URL: The image from the specified URL is read and returned as a ColdFusion image.
- Width and height (`imageType` is optional): A blank ColdFusion image with the specified attributes is returned.
- ColdFusion image variable: An image variable in memory; for example, `#myImage#`.
- A BLOB from a database that contains image data.
- A byte array that contains Base64 image data.
- A Java buffered image. ColdFusion generates an error when the passed attributes cannot create a valid ColdFusion image. The `ImageNew` function and the `{{cfimage}}` read action support the SQL Server Image Column data type. To read Base64 images, use the `ImageReadBase64` function. If the color value is a string, specify a supported named color; see the valid HTML named colors in `cfimage`. For a hexadecimal value, use the form `"##xxxxxx"` or `"xxxxxx"`, where `x` = 0-9 or A-F; use two number signs or none.

⚠️ Note

If you specify the ARGB image type, the image is white; however, if you specify RGB or grayscale, the image is black. To create blank images consistently, use the `canvasColor` parameter.

Example

### Example 1

<!--- Use the ImageNew function to create a 200x200-pixel image in ARGB format. --->

```
<cfset myImage = ImageNew("", 200, 200, "argb")>
<cfimage action="writeTobrowser" source="#myImage#">  
```

Example 2
Example 3

<!--- This example shows how to create a ColdFusion image from a URL. --->
<cfset myImage = ImageNew("http://www.google.com/images/logo_sm.gif")>
<cfset ImageWrite(myImage,"google_via_imagenew.png")>
<img src="google_via_imagenew.png">

Example 4

<!--- This example shows how to use the cffile tag to convert an image file to binary format and pass it as a variable to the ImageNew function. --->
<!---Use the cffile tag to read an image file, convert it to binary format, and write the result to a variable. --->
<cffile action = "readBinary" file = apple.jpg variable = "aBinaryObj"/>
<!--- Use the ImageNew function to create a ColdFusion image from the variable. --->
<cfset myImage = ImageNew(aBinaryObj)>
<!--- This example shows how to use the cffile tag to write a ColdFusion image to a file. --->
<!--- Use the ImageNew function to create a ColdFusion image from a JPEG file. --->
<cfset myImage = ImageNew("../cfdocs/images/artgallery/aiden01.jpg")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Resize the image. --->
<cfset ImageResize(myImage,"50%","")>
<!--- Pass the image object to the cffile tag and write the result to a file on the local drive. --->
<cffile file="#myImage#" action="write" output="c:\test_myImage.jpg">
<cfimage action="writeToBrowser" source="#myImage#">

Example 6

<!--- This example uses cfscript to pass a Java buffered image to the ImageNew function. --->
<cfscript>
bufferedImage = createObject("java", "java.awt.image.BufferedImage");
bufferedImage.init(JavaCast("int", 100), JavaCast("int", 100),
BufferedImage.TYPE_4BYTE_ABGR);
myImage = imageNew(bufferedImage);
</cfscript>
ImageOverlay

Description

Reads two source ColdFusion images and overlays the second source image on the first source image.

Returns

Nothing.

Category

Image functions

Function syntax

```
ImageOverlay(source1, source2 [, rule, alpha])
```

See also

*ImageCopy, ImagePaste, ImageSetAntialiasing, IsImageFile*

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source1</td>
<td>Required. The ColdFusion image that is the bottom layer in the ColdFusion image.</td>
</tr>
<tr>
<td>source2</td>
<td>Required. The ColdFusion image that is the top layer (overlaid on the source1 image) in the ColdFusion image.</td>
</tr>
<tr>
<td>rule</td>
<td>Optional. The following are the supported values: SRC, DST_IN, DST_OUT, DST_OVER, SRC_IN, SRC_OVER, or SRC_OUT. For details, see <em>Java documentation</em>.</td>
</tr>
<tr>
<td>alpha</td>
<td>Optional. The percentage value of transparency.</td>
</tr>
</tbody>
</table>

Usage

The destination image always has the same bounding rectangle as the first source image and the same image type as the two sources. If the two source images do not intersect, the destination image is the same as the first source image. The two source images must have the same color models. For example, you can overlay an RGB image over another RGB image, but you cannot overlay an RGB image on a grayscale image. To verify the color model of an image, use the *ImageInfo* function. Use the *ImageSetAntialiasing* function to improve the quality of the rendered image.
Example

<!--- This example shows how to overlay a smaller image on a larger image. --->
<!--- Create a ColdFusion image from an existing JPEG file and enlarge it by 150%. This image is displayed in the background. --->
<cfimage source="../cfdocs/images/artgallery/maxwell01.jpg" name="myImage" action="resize" width="150%" height="150%">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Create a ColdFusion image from an existing JPEG file. This image is overlaid on the background image. --->
<cfimage source="../cfdocs/images/artgallery/viata05.jpg" name="topImage">
<!--- Overlay the top image on the background image. --->
<cfset ImageOverlay(myImage,topImage)>
<!--- Display the combined image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
**ImagePaste**

**Description**

Takes two images and an (x,y) coordinate and draws the second image over the first image with the upper-left corner at coordinate (x,y).

**Returns**

None

**Category**

*Image functions*

**Function syntax**

```c
<!---- image1 gets modified --->
ImagePaste(image1, image2, x, y)
```

**See also**

*ImageCopy, ImageOverlay, ImageSetAntialiasing, IsImageFile*

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image1</td>
<td>Required. The bottom ColdFusion image.</td>
</tr>
<tr>
<td>Image2</td>
<td>Required. The ColdFusion image that is pasted on top of image1.</td>
</tr>
<tr>
<td>x</td>
<td>Required. The x coordinate where the upper-left corner of image2 is pasted.</td>
</tr>
<tr>
<td>y</td>
<td>Required. The y coordinate where the upper-left corner of image2 is pasted.</td>
</tr>
</tbody>
</table>

**Usage**

Use the *ImageSetAntialiasing* function to improve the quality of the rendered image.

**Example**

```c
```
This example shows how to copy a small rectangular area of one image and paste it over a larger image.

Create a ColdFusion image from an existing JPEG file and name it "myImage1".

Create a ColdFusion image from an existing JPEG file and name it "myImage2".

Copy a rectangular region of myImage1.

Paste the rectangular area over myImage2.

Write the second ColdFusion image to result.jpg.

Display the two source images and the new image.
**ImageRead**

**Description**

Reads the on-disk or in-memory source pathname or URL and creates a ColdFusion image.

**Returns**

A ColdFusion image.

**Category**

Image functions

**Function syntax**

```
ImageRead(path)
```

**History**

ColdFusion 8: Added this function.

**See also**

cfimage, ImageNew, ImageReadBase64, ImageWrite, IsImageFile

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Required. On-disk or in-memory pathname or URL of the source image.</td>
</tr>
</tbody>
</table>

**Usage**

The `ImageRead` function performs the same operation as the `cfimage` read action. However, you cannot use the `cfimage` tag to read and create a ColdFusion image variable in the `cfscript` tag. Use the `ImageRead` function within the `cfscript` tag to read ColdFusion images. The following example reads the image file aiden01.jpg into a variable called myImage and displays the image in the browser:

```cfc
<cfset myImage=ImageRead("../cfdocs/images/artgallery/aiden01.jpg")>
<cfimage action="writeToBrowser" source="#myImage#">
```

For a list of valid image formats, see the supported image file formats listed in `cfimage`. To retrieve a list of readable formats on the server where the ColdFusion application is deployed, use the `GetReadableImageFormats` function.

**Example**
<!--- This example shows how to create a script that reads an image from a URL. --->

<cfscript>
myImage=ImageRead("http://www.google.com/images/logo.gif");
ImageWrite(myImage,"google-logo.gif");
</cfscript>
<p>This image has been downloaded by ColdFusion:</p>
<img src="google-logo.gif" />
<p>This is the original image:</p>
<img src="http://www.google.com/images/logo.gif"/>
ImageReadBase64

Description

Creates a ColdFusion image from a Base64 string.

Returns

A ColdFusion image.

Category

Image functions

Function syntax

ImageReadBase64(string)

See also

ImageNew, ImageRead, ImageWrite, ImageWriteBase64, BinaryDecode, BinaryEncode, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Required. The ColdFusion variable or Base64 string.</td>
</tr>
</tbody>
</table>

Usage

Base64 is a way to describe binary data as a printable string of characters. The ImageReadBase64 function takes Base64 strings as input and creates images from the strings. The strings can be with or without the headers used to pass Base64 images to an HTML `<img src>` tag. Use this function to convert any Base64 string to a ColdFusion image. Some databases store images as Base64 strings instead of BLOB data. You can query the database and use the ImageReadBase64 function to convert the string into a ColdFusion image. This eliminates the intermediary step of converting images with the BinaryEncode and BinaryDecode functions. Really Simple Syndication (RSS) feeds transfer images in the form of embedded Base64 strings in the XML file. Use the ImageReadBase64 function to read these images in ColdFusion.

Example

Example 1{{<!--- This example shows how to read a Base64 string with headers. --->}}

```
<cfset myImage = ImageReadBase64("data:image/jpg;base64,/9j/4AAQSkZJRgABAQ..............")>
<cfimage source="#myImage#" destination="test_my64.jpeg" action="write">
```
Example 2

<!--- This example shows how to read a Base64 string without headers. --->

```coldfusion
<cfset myImage = ImageReadBase64("/9j/4AAQSkJRgABAQ.................")>
<cfimage source="#myImage#" destination="test_my64.jpeg" action="write"/>
```
ImageResize

Description

Resizes a ColdFusion image.

Returns

Nothing.

Category

Image functions

Function syntax

```
ImageResize(name, width, height [, interpolation, blurFactor])
```

See also

cfimage, ImageSetAntialiasing, ImageScaleToFit, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>width</td>
<td>Required. New width of the ColdFusion image. If this value is blank, the width is calculated proportionately to the height.</td>
</tr>
<tr>
<td>height</td>
<td>Required. New height of the ColdFusion image. If this value is blank, the height is calculated proportionately to the width.</td>
</tr>
</tbody>
</table>
### interpolation

Optional. The interpolation method for resampling. You can specify a specific interpolation algorithm by name (for example, hamming), by image quality (for example, mediumQuality), or by performance (for example, highestPerformance). Valid values are:

- `highestQuality` (default)
- `highQuality`
- `mediumQuality`
- `highestPerformance`
- `highPerformance`
- `mediumPerformance`
- `nearest`
- `bilinear`
- `bicubic`
- `bessel`
- `blackman`
- `hamming`
- `hanning`
- `hermite`
- `lanczos`
- `mitchell`
- `quadratic`

See Interpolation algorithms in the Usage section for more information.

### blurFactor

Optional. The blur factor used for resampling. The higher the blur factor, the more blurred the image (also, the longer it takes to resize the image). This value must be from 1 through 10.

### Usage

You can use this function to enlarge an image or create a thumbnail image. To specify the height or width in pixels, enter the integer, for example, 100. To specify the height or width as a percentage, enter the percentage followed by the percent symbol, for example, 50%. To resize an image by one dimension (for example, height), specify the height and leave width value blank (" "). ColdFusion calculates the width proportionally to the height. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

### Interpolation algorithms

Interpolation algorithms let you fine-tune how images are resampled. Each algorithm balances image quality against performance: in general, the higher the image quality, the slower the performance. Quality and performance differ based on image type and the size of the source file. The following table describes the algorithms and their named equivalents based on average test results:

<table>
<thead>
<tr>
<th>Value</th>
<th>Named equivalents</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>highestQuality (default)</td>
<td>lanczos</td>
<td>Highest image quality with low performance</td>
</tr>
<tr>
<td>Lookup Name</td>
<td>Filter Name(s)</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>highQuality, mediumPerformance</td>
<td>mitchell, quadratic</td>
<td>Good image quality with slightly higher performance</td>
</tr>
<tr>
<td>mediumQuality, highPerformance</td>
<td>hamming, hanning, hermite</td>
<td>Medium quality image with medium performance</td>
</tr>
<tr>
<td></td>
<td>blackman, bessel</td>
<td>Slightly distorted image quality with high performance</td>
</tr>
<tr>
<td>highestPerformance</td>
<td>nearest, bicubic, bilinear</td>
<td>Poor image quality with highest performance</td>
</tr>
</tbody>
</table>

Example

```coldfusion
<!--- This example shows how to resize an image to 50% of original size and resize it proportionately to the new width. Notice that the height is blank. --->
<cfset myImage=ImageNew("http://www.google.com/images/logo_sm.gif")>
<cfset ImageResize(myImage,"50%","","blackman",2)>
<!--- Save the modified image to a file called "test_myImage.jpeg" and display the image in a browser. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpeg" overwrite="yes">
<!--- Display the source image and the thumbnail image. --->
<img src="http://www.google.com/images/logo_sm.gif"/>
<img src="test_myImage.jpeg"/>
```
ImageRotate

Description
Rotates a ColdFusion image at a specified point by a specified angle.

Returns
Nothing.

Category
Image functions

Function syntax
ImageRotate(name, angle [, x, y, interpolation])

See also
cfimage, ImageFlip, ImageSetAntialiasing, IsImageFile

History
ColdFusion 8: Added this function.

Parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>angle</td>
<td>Required. The rotation angle in degrees.</td>
</tr>
<tr>
<td>x</td>
<td>Optional. The x coordinate for the point of rotation. The default value is 2.</td>
</tr>
<tr>
<td>y</td>
<td>Optional. The y coordinate for the point of rotation. The default value is 2.</td>
</tr>
<tr>
<td>interpolation</td>
<td>Optional. Type of interpolation:</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>nearest:</td>
<td>Applies the nearest neighbor method of interpolation. Image quality is lower than the other interpolation methods, but processing is fastest (default).</td>
</tr>
<tr>
<td>bilinear:</td>
<td>Applies the bilinear method of interpolation. The quality of the image is less pixelated than the default, but processing is slower.</td>
</tr>
<tr>
<td>bicubic:</td>
<td>Applies the bicubic method of interpolation. Generally, the quality of image is highest with this method and processing is slowest.</td>
</tr>
</tbody>
</table>

Usage

Specify both the x and the y coordinates or neither. If you do not specify the x and y coordinates, the point of rotation is the center of the image, which is the default position. Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

Example

Example 1

```cfc
<!--- This example shows how to rotate an image by 10 degrees. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfset myImage=ImageNew("../cfdocs/images/artgallery/jeff05.jpg")>
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Rotate the image by 10 degrees. --->
<cfset ImageRotate(myImage,10)>
<cfimage source="#myImage#" action="writeToBrowser">
```

Example 2

```cfc
<!--- This example shows how to rotate an image by 10 degrees and change the interpolation to bicubic for higher resolution. The image is rotated at the (10,90) coordinates. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<cfset ImageRotate(myImage,10,10,90,"bicubic")>
<cfimage source="#myImage#" destination="testMyImage.jpeg" action="write" overwrite="yes">
<img src="../cfdocs/images/artgallery/jeff05.jpg">
<img src="testMyImage.jpeg">
```
ImageRotateDrawingAxis

Description
Rotates all subsequent drawing on a ColdFusion image at a specified point by a specified angle.

Returns
A ColdFusion image.

Category
Image functions

Function syntax

ImageRotateDrawingAxis(name, angle [, x, y])

See also

History
ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>angle</td>
<td>Required. The rotation angle in degrees.</td>
</tr>
<tr>
<td>x</td>
<td>Optional. The x coordinate for the point of rotation. The default value is 0.</td>
</tr>
<tr>
<td>y</td>
<td>Optional. The y coordinate for the point of rotation. The default value is 0.</td>
</tr>
</tbody>
</table>

Usage
The default position of the origin is 0,0. To revert to the original drawing axis, call the same (x,y) parameters with the negative of the original angle. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example
<!--- This example shows how to create an image with three shapes drawn on the same axis. --->
<!--- Use ImageNew to create a 300x300-pixel image. --->
<cfset myImage=ImageNew("",300,300)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the drawing axis to 30 degrees and the point of rotation at (10,10). --->
<cfset ImageRotateDrawingAxis(myImage,30,10,10)>
<!--- Set the drawing color to blue. --->
<cfset ImageSetDrawingColor(myImage,"blue")>
<!--- Draw three shapes with the same color and drawing axis. --->
<cfset ImageDrawRect(myImage,150,10,10,150,"yes")>
<cfset ImageDrawOval(myImage,200,40,45,65,"yes")>
<cfset ImageDrawRect(myImage,275,10,10,150,"yes")>
<cfimage source="#myImage#" action="writeToBrowser">
ImageScaleToFit

Description

Creates a resized image with the aspect ratio maintained.

Returns

Nothing.

Category

Image functions

Function syntax

```
ImageScaleToFit(name, fitWidth, fitHeight [, interpolation , blurFactor])
```

See also

cfimage, ImageResize, ImageSetAntialiasing, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>fitWidth</td>
<td>Required. The width of the bounding box in pixels. You can specify an integer, or an empty string (&quot;&quot;) if the fitHeight is specified. See the Usage section for more information.</td>
</tr>
<tr>
<td>fitHeight</td>
<td>Required. The height of the bounding box in pixels. You can specify an integer, or an empty string (&quot;&quot;) if the fitWidth is specified. See the Usage section for more information.</td>
</tr>
</tbody>
</table>
### interpolation

Optional. The interpolation method for resampling. You can specify a specific interpolation algorithm by name (for example, `hamming`), by image quality (for example, `mediumQuality`), or by performance (for example, `highestPerformance`). Valid values are:

- `highestQuality` (default)
- `highQuality`
- `mediumQuality`
- `highestPerformance`
- `highPerformance`
- `mediumPerformance`
- `nearest`
- `bilinear`
- `bicubic`
- `bessel`
- `blackman`
- `hamming`
- `hanning`
- `hermite`
- `lanczos`
- `mitchell`
- `quadratic`

See *Interpolation algorithms* in the Usage section for more information.

### blurFactor

Optional. The blur factor used for resampling. The higher the blur factor, the more blurred the image (also, the longer it takes to resize the image). Valid values are 1-10.

## Usage

Use this operation to resize images or create thumbnail images while maintaining the aspect ratio. Specify the `fitWidth` and `fitHeight` parameters; either the `fitWidth` or the `fitHeight` can be an empty string:

```cfset
ImageScaleToFit(myImage, 100, "")
```

In this example, the `ImageScaleToFit` function resizes the image so that it fits in a 100×100-pixel square; the width of the resulting image is 100 pixels and the height is less than or equal to 100 pixels. For example, if the source image is 400×200 pixels, the resulting image is 100×50 pixels. Likewise, if you specify the `fitHeight` parameter and an empty string for the `fitWidth` parameter, the `ImageScaleToFit` function resizes the image so that the height equals the `fitHeight` parameter and the width of the image is scaled proportionately:

```cfset
ImageScaleToFit(myImage, ",", 100)
```

In this example, a 400×200-pixel source image is resized to 200×100 pixels, and a 200×400-pixel image is resized to...
50x100 pixels. If you set both the fitWidth and the fitHeight parameters, the ImageScaleToFit function resizes the image proportionately so that both conditions are true: the width of the resulting image is less than or equal to the fitWidth, and the height is less than or equal to the fitHeight:

```coldfusion
<cfset ImageScaleToFit(myImage,100,200)>
```

In this example, a 400x200-pixel source image is resized to 100x50 pixels, and a 200x400-pixel source image is resized to 100x200 pixels. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

Example

```coldfusion
<!--- This example shows how to resize an image to fit a 100x100-pixel square while maintaining the aspect ratio. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/jeff05.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<cfset ImageScaleToFit(myImage,100,"","lanczos")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```
**ImageSetAntialiasing**

**Description**

Switches antialiasing on or off in rendered graphics.

**Returns**

Nothing.

**Category**

*Image functions*

**Function syntax**

```
ImageSetAntialiasing(name [, antialias])
```

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>antialias</td>
<td>Optional. Antialiasing switch:</td>
</tr>
<tr>
<td></td>
<td>• on (default)</td>
</tr>
<tr>
<td></td>
<td>• off</td>
</tr>
</tbody>
</table>

**Usage**

The `ImageSetAntialiasing` function is used to turn antialiasing on and off when drawing shapes and text in an image. Antialiasing is a technique used to soften jagged edges. Turn on antialiasing when using other image functions, such as `ImageDrawRoundRect` and `ImageRotate`, to improve the quality of the rendered image. Notice that antialiasing decreases performance.

**Example**

Example 1
Example 1

<!--- This example shows how to turn off antialiasing for a ColdFusion image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfset myImage=ImageNew("../cfdocs/images/artgallery/elecia02.jpg")>
<!--- Turn off antialiasing. --->
<cfset ImageSetAntialiasing(myImage,"off")>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">

Example 2

<!--- This example shows how to turn on antialiasing to improve the output of the ImageDrawRoundRect function. --->
<!--- Create a 200x200-pixel image. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Set the drawing color for the image to blue. --->
<cfset ImageSetDrawingColor(myImage,"blue")>
<!--- Turn on antialiasing. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Draw a blue filled square with round corners of arcWidth=10 and arcHeight=2. --->
<cfset ImageDrawRoundRect(myImage,5,5,190,190,"yes",10,2)>
<!--- Rotate the image 30 degrees. --->
<cfset ImageRotate(myImage,30)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
ImageSetBackgroundColor

Description

Sets the background color for the ColdFusion image. The background color is used for clearing a region. Setting the background color only affects the subsequent ImageClearRect calls.

Returns

Nothing.

Category

Image functions

Function syntax

ImageSetBackgroundColor(name, color)

See also

ImageClearRect, ImageSetAntialiasing, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>
| color     | Required. Background color:  
  • Hexadecimal value of RGB color. For example, specify the color white as ##FFFFFF or FFFFFF.  
  • String value of color (for example, "black", "red", "green"). The default value of the drawing color is "black".  
  • List of three numbers for (R,G,B) values. Each value must be in the range 0-255. |

Usage

If the color value is a string, specify a supported named color; see the list of valid HTML named colors in cffimage. For a hexadecimal value, use the form "##xxxxxx" or "xxxxxxx", where x = 0-9 or A-F; use two number signs or none. Use this function in conjunction with the ImageClearRect function to clear a rectangular area of an image and set it to a specified color.

Example
<!--- This example shows how to set the background color, and then draw a rectangle on an image filled with that color. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage name="myImage" source="../cfdocs/images/artgallery/maxwell01.jpg">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Set the background color to magenta. --->
<cfset ImageSetBackgroundColor(myImage,"magenta")>
<!--- Clear the rectangle specified on myImage with the background color specified for the image. --->
<cfset ImageClearRect(myImage,36,45,100,100)>
<!--- Display the modified image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
ImageSetDrawingColor

Description

Sets the current drawing color for ColdFusion images. All subsequent graphics operations use the specified color.

Returns

Nothing.

Category

Image functions

Function syntax

ImageSetDrawingColor(name, color)

See also

ImageSetAntialiasing, ImageSetBackgroundColor, ImageSetDrawingStroke, ImageSetDrawingTransparency, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>color</td>
<td>Required. Drawing color:</td>
</tr>
<tr>
<td></td>
<td>• Hexadecimal value of RGB color. For example, specify the color white as &quot;##FFFFFF&quot; or &quot;FFFFFF&quot;.</td>
</tr>
<tr>
<td></td>
<td>• String value of color (for example, &quot;black&quot;, &quot;red&quot;, &quot;green&quot;). The default value of the drawing color is &quot;black&quot;.</td>
</tr>
<tr>
<td></td>
<td>• List of three numbers for (R,G,B) values. Each value must be in the range 0-255.</td>
</tr>
</tbody>
</table>

Usage

Use the ImageSetDrawingColor function to control the color of all subsequent drawing objects in ColdFusion images. For example, you can use this function to set the drawing color to red once, and then draw a circle, a square, and 10 lines in that color. If the color value is a string, specify a supported named color; see the list of valid HTML named colors in cfimage. For a hexadecimal value, use the form "##xxxxxx" or "xxxxxx", where x = 0-9 or A-F; use two number signs or none.
Example

```coldfusion
<!--- This example shows how to set the drawing color and draw several objects in that color. --->
<!--- Use the ImageNew function to create a ColdFusion image. --->
<cfset myImage=ImageNew("",200,300)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Set the drawing color to pink. --->
<cfset ImageSetDrawingColor(myImage,"pink")>
<!--- Draw a pink line. --->
<cfset ImageDrawLine(myImage,1,1,200,300)>
<!--- Draw a pink oval. --->
<cfset ImageDrawOval(myImage,40,50,80,40)>
<!--- Draw another pink oval. --->
<cfset ImageDrawOval(myImage,15,180,80,40)>
<!--- Draw a pink rectangle. --->
<cfset ImageDrawRect(myImage,100,100,45,65)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```
ImageSetDrawingStroke

Description

Sets the drawing stroke for points and lines in subsequent ColdFusion images.

Returns

Nothing.

Category

Image functions

Function syntax

ImageSetDrawingStroke(name [, attributeCollection])

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>attributeCollection</td>
<td>Optional. The structure used to specify the line attributes. See the Usage section.</td>
</tr>
</tbody>
</table>

Usage

Use the ImageSetDrawingStroke function to control the line attributes of all subsequent drawing objects in a ColdFusion image. For example, you can use this function to set the drawing stroke to a dash pattern once, and then create a rectangle, two ovals, and five lines with that pattern. If a blank or no attribute structure is passed, the drawing stroke is reset to the default values.

attributeCollection

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>Pen width, which is measured perpendicularly to the pen trajectory.</td>
</tr>
</tbody>
</table>
| endcaps       | Decoration applied to the ends of unclosed subpaths and dash segments. Subpaths that start and end on the same point are considered unclosed if they do not have a close segment:  
|              | • butt  
|              | • round  
|              | • square |
| lineJoins    | Type of line joins:  
|              | • bevel  
|              | • miter  
|              | • join |
| miterLimit   | The limit to trim a line join that has a mitered join decoration. (Use only when lineJoins = "miter"). A line join is trimmed when the ratio of miter length to stroke width is greater than the miterLimit value. The miter length is the diagonal length of the miter, which is the distance between the inside corner and the outside corner of the intersection. The smaller the angle formed by two line segments, the longer the miter length and the sharper the angle of intersection. The default value is 10.0, which trims all angles less than 11 degrees. Trimming miters converts the decoration of the line join to bevel. |
| dashArray    | An array of numbers that indicates the dash pattern. For example, if dashArray is (8,4), the dash pattern is 8 pixels solid, 4 pixels blank, 8 pixels solid, 4 pixels blank, and so on. |
| dash_phases  | An offset into the dash pattern. For example, a dash_phase of 2, and a dashArray of (8,4) produces the dash pattern of 6 pixels solid, 4 pixels blank, 8 pixels solid, 4 pixels blank, and so on. |

**Example**

**Example 1**
<!-- This example shows how to create an attribute collection for the ImageSetDrawingStroke function and draws a line with those attributes. -->

<!-- Use the ImageNew function to create a ColdFusion image. -->
<cfset myImage=ImageNew("",200,200)>

<!-- Create an attribute collection to pass to the ImageSetDrawingStroke function. Create a stroke that is 10-pixels wide, has round endcaps, and has a dash pattern of (8,4). -->
<cfset attr = StructNew()>
<cfset attr.width = 2>
<cfset attr.endcaps = "round">
<cfset dashPattern = ArrayNew(1)>
<cfset dashPattern[1] = 8>
<cfset dashPattern[2] = 4>
<cfset attr.dashArray = dashPattern>

<!-- Apply the attribute collection to the ImageSetDrawingStroke function for the image. -->
<cfset ImageSetDrawingStroke(myImage,attr)>

<!-- Draw a line on the ColdFusion image with the drawing stroke attributes. -->
<cfset ImageDrawLine(myImage,20,20,40,150)>

<!-- Display the image in a browser. -->
<cfimage source="#myImage#" action="writeToBrowser">
<cfset myImage=ImageNew("",500,500)>
<cfset ImageSetDrawingColor(myImage,"cyan")>
<cfset ImageDrawLine(myImage,30,40,200,190)>
<cfset attr = StructNew()>
<cfset attr.width = 1>
<cfset attr.endcaps = "round">
<cfset dashPattern = ArrayNew(1)>
<cfset dashPattern[1] = 3>
<cfset dashPattern[2] = 4>
<cfset dashPattern[3] = 8>
<cfset attr.dashArray = dashPattern>
<cfset ImageSetDrawingStroke(myImage,attr)>
<cfset ImageSetDrawingColor(myImage,"yellow")>
<cfset ImageDrawRect(myImage,200,50,210,200)>
<cfset ImageSetDrawingStroke(myImage)>
<cfset ImageSetDrawingColor(myImage,"green")>
<cfset ImageDrawQuadraticCurve(myImage,120,320,5,15,380,280)>
<cfimage source="#myImage#" action="writeToBrowser"
**ImageSetDrawingTransparency**

**Description**

Specifies the degree of transparency of drawing functions.

**Returns**

Nothing.

**Category**

Image functions

**Function syntax**

```plaintext
ImageSetDrawingTransparency(name, percent)
```

**See also**

ImageSetAntialiasing, ImageSetBackgroundColor, ImageSetDrawingColor, ImageSetDrawingStroke, IsImageFile

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
</tbody>
</table>
| percent   | Required. Percent of transparency:  
- 0 = opaque  
- 100 = transparent  
Decimal values are valid. |

**Usage**

By default drawing images are opaque. Use this function to create watermarks or other translucent images. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

**Example**

Example 1
Example 2

<!---- This example shows how to create a watermark from the a JPEG file. --->
<!---- Create a ColdFusion image from a JPEG file. --->
<cfimage source="../cfdocs/getting_started/photos/somewhere.jpg" name="myImage" action="read">
<!---- Set the drawing transparency to 75%. --->
<cfset ImageSetDrawingTransparency(myImage,75)>
<!---- Create a ColdFusion image from a picture in the cfartgallery. --->
<cfimage source="../cfdocs/images/artgallery/raquel05.jpg" name="myImage2" action="read">
<!---- Set the drawing transparency to 30%. --->
<cfset ImagesetDrawingTransparency(myImage,30)>
<!---- Paste the ColdFusion log over the picture at coordinates (0,0).--->
<cfset ImagePaste(myImage,myImage2,0,0)>
<!---- Display the two source images and the result. --->
<cfimage source="#myImage#" destination="watermark.jpg" action="write" overwrite="yes">
<img src="../cfdocs/getting_started/photos/somewhere.jpg">
<img src="../cfdocs/images/artgallery/raquel05.jpg">
<img src="watermark.jpg">

Example 3
<!--- This code creates a ColdFusion image to be used as a watermark. --->
<cfimage action="read" name="logo" source="../cfdocs/getting_started/photos/somewhere.jpg">
<cfset imageGrayscale(logo)>
<cfset imageRotate(logo,45)>
<!--- This code creates the ColdFusion image to be used as the base image. --->
<cfimage action="read" source="../cfdocs/images/artgallery/raquel05.jpg" name="baseImage">
<!--- This code sets the drawing transparency for the base image to 80%. --->
<cfset ImageSetDrawingTransparency(baseImage,80)>
<!--- This code pastes the watermark image onto the base image at the coordinates (0,0). --->
<cfset ImagePaste(baseImage,logo,0,0)>
<!--- This code writes the result to a file. --->
<cfimage action="write" source="#baseImage#" destination="abc_watermark.jpg" overwrite="yes">
<!--- This code displays the image used as a watermark and the result. --->
<img src="../cfdocs/getting_started/photos/somewhere.jpg"/>
<img src="abc_watermark.jpg"/>
**ImageSharpen**

**Description**

Sharpen a ColdFusion image by using the unsharp mask filter.

**Returns**

Nothing.

**Category**

Image functions

**Function syntax**

```
ImageSharpen(name [, gain])
```

**See also**

ImageBlur, ImageSetAntialiasing, IsImageFile

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>gain</td>
<td>Optional. -1 &lt;= gain &lt;= 2. Gain values can be integers or real numbers. The default value is 1.0. The value determines whether the image is blurred or sharpened:</td>
</tr>
<tr>
<td></td>
<td>• If &gt; 0, the image is sharpened.</td>
</tr>
<tr>
<td></td>
<td>• If = 0, no effect</td>
</tr>
<tr>
<td></td>
<td>• If &lt; 0, the image is blurred.</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to sharpen outlines in photographs. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

**Example**
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul01.jpg" name="myImage">

<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>

<!--- Sharpen myImage by 2. --->
<cfset ImageSharpen(myImage,2)>

<!--- Write the sharpened image to a file. --->
<cfimage source="#myImage#" action="write" destination="test_myImage.jpg" overwrite="yes">

<!--- Display the original and the sharpened images. --->
<img src="../cfdocs/images/artgallery/paul01.jpg"/>
<img src="test_myImage.jpg"/>
**ImageShear**

**Description**

Shears an image either horizontally or vertically. For each pixel \((x, y)\) of the destination, the source value at the fractional subpixel position \((x', y')\) is constructed with an Interpolation object and written to the destination.

**Returns**

Nothing.

**Category**

*Image functions*

**Function syntax**

```
ImageShear(name, shear [, direction, interpolation])
```

**See also**

*ImageSetAntialiasing, ImageShearDrawingAxis, IsImageFile*

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>name</code></td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td><code>shear</code></td>
<td>Required. Shear value. Coordinates can be integers or real numbers.</td>
</tr>
<tr>
<td><code>direction</code></td>
<td>Optional. Shear direction:</td>
</tr>
<tr>
<td></td>
<td>• horizontal (default)</td>
</tr>
<tr>
<td></td>
<td>• vertical</td>
</tr>
</tbody>
</table>
### interpolation

**Optional. Type of interpolation:**
- **nearest:** Applies the nearest neighbor method of interpolation. Image quality is lower than the other interpolation methods, but processing is fastest (default).
- **bilinear:** Applies the bilinear method of interpolation. The quality of the image is less pixelated than the default, but processing is slower.
- **bicubic:** Applies the bicubic method of interpolation. Generally, the quality of image is highest with this method and processing is slowest.

### Usage

Use this function to distort an image. If the `direction` parameter is set to `horizontal`, \( x' = (x - y \times \text{shear}) \) and \( y' = y \). If the `direction` parameter is set to `vertical`, \( x' = x \) and \( y' = (y - x \times \text{shear}) \). Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

### Example

```cfml
<!--- This example shows how to shear an image. --->
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Shear the image by a factor of 1 on a horizontal axis. --->
<cfset ImageShear(myImage,1,"horizontal")>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```
**ImageShearDrawingAxis**

**Description**
Shears the drawing canvas.

**Returns**
Nothing.

**Category**
Image functions

**Function syntax**

```plaintext
ImageShearDrawingAxis(name, shx, shy)
```

**See also**

`ImageRotateDrawingAxis`, `ImageSetAntialiasing`, `ImageShear`, `IsImageFile`

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>shx</td>
<td>Required. The multiplier by which coordinates are shifted in the positive x axis direction as a function of the y coordinate.</td>
</tr>
<tr>
<td>shy</td>
<td>Required. The multiplier by which coordinates are shifted in the positive y axis direction as a function of the x coordinate.</td>
</tr>
</tbody>
</table>

**Usage**

For each pixel (x,y) of the destination, the source value at the fractional subpixel position (x',y') is constructed with an interpolation object and written to the destination. If the `direction` parameter is equal to horizontal, x' = (x - y'shear) and y' = y. If the `direction` parameter is equal to vertical, x' = x and y' = (y - x'shear). Use the `ImageSetAntialiasing` function to improve the quality of the rendered image.

**Example**
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/paul03.jpg" name="myImage">
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>
<!--- Set the shear drawing axis. --->
<cfset ImageShearDrawingAxis(myImage,0.5,0.5)>
<!--- Draw a rectangle on the image with the shear settings. --->
<cfset ImageDrawRect(myImage,50,50,50,50)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser"
ImageTranslate

Description

Copies an image to a new location on the plane.

Returns

Nothing.

Category

Image functions

Function syntax

```
ImageTranslate(name, xTrans, yTrans [, interpolation])
```

See also

ImageSetAntialiasing, ImageShear, ImageTranslateDrawingAxis, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>xTrans</td>
<td>Required. Displacement in the x direction.</td>
</tr>
<tr>
<td>yTrans</td>
<td>Required. Displacement in the y direction.</td>
</tr>
<tr>
<td>interpolation</td>
<td>Optional. Type of interpolation used for resampling:</td>
</tr>
<tr>
<td></td>
<td>• nearest: Applies the nearest neighbor method of interpolation. Image quality is lower than the other interpolation methods, but processing is fastest (default).</td>
</tr>
<tr>
<td></td>
<td>• bilinear: Applies the bilinear method of interpolation. The quality of the image is less pixelated than the default, but processing is slower.</td>
</tr>
<tr>
<td></td>
<td>• bicubic: Applies the bicubic method of interpolation. Generally, the quality of image is highest with this method and processing is slowest.</td>
</tr>
</tbody>
</table>

Usage
For each pixel \((x, y)\) of the destination, the source value at the fractional subpixel position 
\((x - x_{\text{Trans}}, y - y_{\text{Trans}})\) is constructed with an interpolation object and written to the destination. If both \(x_{\text{Trans}}\) and \(y_{\text{Trans}}\) are integral, the operation wraps the source image to change the image's position in the coordinate plane. Use the \texttt{ImageSetAntialiasing} function to improve the quality of the rendered image.

Example

```coldfusion
<!--- Create a ColdFusion image from an existing JPEG file. --->
<cfimage source="../cfdocs/images/artgallery/aiden01.jpg" name="myImage">

<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage,"on")>

<!--- Offset the image's position to \( (20,10) \). --->
<cfset ImageTranslate(myImage,20,10)>

<!--- Display the offset image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
```
**ImageTranslateDrawingAxis**

**Description**

Translates the origin of the image context to the point \((x,y)\) in the current coordinate system. Modifies the image context so that its new origin corresponds to the point \((x,y)\) in the image's original coordinate system.

**Returns**

Nothing.

**Category**

Image functions

**Function syntax**

\[
\text{ImageTranslateDrawingAxis}(\text{name}, x, y)
\]

**See also**


**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>x</td>
<td>x coordinate</td>
</tr>
<tr>
<td>y</td>
<td>y coordinate</td>
</tr>
</tbody>
</table>

**Usage**

All coordinates used in subsequent rendering operations on this image context are relative to the new origin. Use the ImageSetAntialiasing function to improve the quality of the rendered image.

**Example**
<!--- Create a 200x200-pixel image. --->
<cfset myImage=ImageNew("",200,200)>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Translate the origin to (100,20). --->
<cfset ImageTranslateDrawingAxis(myImage,100,20)>
<!--- Draw a rectangle at the offset location. --->
<cfset ImageDrawRect(myImage,50,60,40,50)>
<!--- Display the image in a browser. --->
<cfimage source="#myImage#" action="writeToBrowser">
ImageWrite

Description

Writes a ColdFusion image to the specified on-disk or in-memory destination.

Returns

Nothing.

Category

Image functions

Function syntax

ImageWrite(name [, destination, quality, overwrite])

See also

cfimage, GetWriteableImageFormats, ImageNew, ImageRead, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>destination</td>
<td>Optional. The absolute or relative on-disk or in-memory pathname where you write the file. If you create the image with the ImageNew function or another operation where you do not specify the filename, specify the destination parameter. The file format is derived from the extension of the filename. The default value for this parameter is the filename of the original image source.</td>
</tr>
<tr>
<td>quality</td>
<td>Optional. Defines the JPEG quality used to encode the image. This parameter applies only to destination files with an extension of JPG or JPEG. Valid values are fractions that range from 0 through 1 (the lower the number, the lower the quality). The default value is 0.75.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Optional. If set to true, overwrites the destination file.</td>
</tr>
</tbody>
</table>

Usage
Use the following syntax to specify an in-memory file, which is not written to disk. In-memory files speed processing of transient data.

```plaintext
ram:///filepath
```

The filepath can include directories, for example `ram:///petStore/images/poodle.jpg`. Create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications. The file format is derived from the file extension, therefore, use this function to convert images. For a list of valid formats to write, see the supported image file formats listed in `cfimage`. To retrieve a list of writable formats on the server where the ColdFusion application is deployed, use the `GetWriteableImageFormats` function.

⚠️ **Note**

Converting images between one file format to another is time-consuming. Also, image quality can degrade; for example, PNG images support 24-bit color, but GIF images support only 256 colors. Converting transparent images (images with alpha) can degrade image quality.

**Example**

```html
<!--- This example shows how to convert a GIF image to a PNG image. --->
<!--- Use the ImageNew function to create a ColdFusion image. --->
<cfset myImage = ImageNew("http://www.google.com/images/logo_sm.gif")>
<!--- Convert the image to a PNG format. --->
<cfset ImageWrite(myImage,"google.png")>
<!--- Display the PNG image. --->
<img src="google.png">
```
ImageWriteBase64

Description

Writes Base64 images to the specified on-disk or in-memory destination.

Returns

Base64 string.

Category

Image functions

Function syntax

```
ImageWriteBase64(name, destination, format [, inHTMLFormat, optional])
```

See also

cfimage, ImageReadBase64, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>destination</td>
<td>Required. The absolute or relative on-disk or in-memory pathname where you write the file.</td>
</tr>
<tr>
<td>format</td>
<td>Required. The format</td>
</tr>
<tr>
<td>inHTMLFormat</td>
<td>Optional. Specify whether Base64 output includes the headers used by the Base64 images in the HTML <code>&lt;img&gt;</code> tag:</td>
</tr>
<tr>
<td></td>
<td>• yes</td>
</tr>
<tr>
<td></td>
<td>• no (default)</td>
</tr>
<tr>
<td>overwrite</td>
<td>Optional. If set to true, overwrites the destination file.</td>
</tr>
</tbody>
</table>

Usage

You use the `ImageWriteBase64` function to encode image data as a string of printable characters. This is useful for several applications, including sending images by e-mail and storing images in database text fields. Use the following syntax to specify an in-memory file, which is not written to disk. In-memory files speed processing of
transient data.

```
ram:///filepath
```

The filepath can include directories, for example `ram:///petStore/images/poodle.jpg`. Create the directories in the path before you specify the file. For more information on using in-memory files, see Working with in-memory files in Optimizing ColdFusion applications in the Developing ColdFusion Applications. If you do not specify a file format, ColdFusion cannot recognize the format required to encode the image before converting to Base64, and generates errors. You can verify whether ColdFusion reads a Base64 string properly in the following ways:

- Use the cfdump tag. For example: `<cfdump var="#myImage#">`
- Use the ImageInfo function. For example: `<cfset ImageInfo(myImage)>`
- Use the ImageWrite function and save the image as a JPEG file. Then open the JPEG file in a browser or imaging application.

Example

```
<!---- This example shows how to convert a JPEG image to Base64 format and save it to a file. --->
<!---- Create a new ColdFusion image. --->
<cfset myImage=ImageNew("../cfdocs/images/artgallery/jeff01.jpg")>
<!---- Convert the image to Base64 format and write it to a file. --->
<cfset ImageWriteBase64(myImage,"jeffBase64.txt","jpg","yes")>
```
ImageXORDrawingMode

Description

Sets the paint mode of the image to alternate between the image's current color and the new specified color.

Returns

Nothing.

Category

Image functions

Function syntax

ImageXORDrawingMode(name, c1)

See also

ImageSetDrawingColor, IsImageFile

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion image on which this operation is performed.</td>
</tr>
<tr>
<td>c1</td>
<td>Required. XOR alternation color:</td>
</tr>
<tr>
<td></td>
<td>• Hexadecimal value of the RGB color. For example, specify the color white as &quot;##FFFFFF&quot; or &quot;FFFFFF&quot;.</td>
</tr>
<tr>
<td></td>
<td>• String value of color (for example, &quot;black&quot;, &quot;red&quot;, &quot;green&quot;).</td>
</tr>
</tbody>
</table>

Usage

This function alternates pixels between the current color and a new specified XOR (exclusive Or) color. When drawing operations are performed, pixels that are the current color are changed to the specified color, and vice versa. Pixels that are of colors other than current color or the new specified color are changed in an unpredictable but reversible manner. If the same figure is drawn twice, all pixels are restored to their original values. If the color value is a string, specify a supported named color; see the list of valid HTML named colors in cffimage. For a hexadecimal value, use the form "##xxxxxx" or "xxxxxx", where x = 0-9 or A-F; use two number signs or none.

Example
<!--- This example shows how to draw rectangles with alternating colors where they overlap. --->
<!--- Use the ImageNew function to create a 300x200-pixel image in RGB format. --->
<cfset myImage = ImageNew("",300,200,"rgb")>
<!--- Turn on antialiasing to improve image quality. --->
<cfset ImageSetAntialiasing(myImage)>
<!--- Set the drawing color to white. --->
<cfset ImageSetDrawingColor(myImage,"white")>
<!--- Draw a white filled rectangle that is the size of the original image. --->
<cfset ImageDrawRect(myImage,0,0,300,200,"yes")>
<!--- Set the XOR drawing mode to white. --->
<cfset ImageXORDrawingMode(myImage,"white")>
<!--- Set the drawing color to red. --->
<cfset ImageSetDrawingColor(myImage,"red")>
<!--- Draw a filled red rectangle. --->
<cfset ImageDrawRect(myImage,50,50,150,100,"yes")>
<!--- Translate the drawing axis to (25,25). --->
<cfset ImageTranslateDrawingAxis(myImage,25,25)>
<!--- Set the drawing color to blue. --->
<cfset ImageSetDrawingColor(myImage,"blue")>
<!--- Draw a blue filled rectangle at the offset location. --->
<cfset ImageDrawRect(myImage,50,50,150,100,"yes")>
<!--- Save the ColdFusion image as a PNG file. --->
<cfset ImageWrite(myImage,"xortest.png")>
<!--- Display the PNG file. --->
<img src="xortest.png">
ImageGetMetadata

You can use this function to retrieve all metadata about the image including exif, IPTC, camera-specific information.

Returns

Returns a struct containing the information about the image.

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>Required. An Image object.</td>
</tr>
</tbody>
</table>

Example

```
<cfimage action="read" name="myImage" source="../MasterImages/sky.jpg">
<cfsset getMetaData = ImageGetMetadata(myImage)>
  <cfoutput>
    AF Area Mode : #StructFind(All,"AF Area Mode")#
    Camera Uptime : #StructFind(All,"Camera Uptime")#
    Exif Version : #StructFind(All,"Exif Version")#
  </cfoutput>
```
Functions in-k

- IncrementValue
- InputBaseN
- Insert
- Int
- Invoke
- IsArray
- IsAuthenticated
- IsAuthorized
- IsBinary
- IsBoolean
- IsClosure
- IsCustomFunction
- IsDate
- IsDDX
- IsDebugMode
- IsDefined
- IsImage
- IsImageFile
- IsInstanceOf
- IsIPv6
- IsJSON
- IsK2ServerABroker
- IsK2ServerDocCountExceeded
- IsK2ServerOnline
- IsLeapYear
- IsLocalHost
- IsNull
- IsNumeric
- IsNumericDate
- IsObject
- isOnline
- IsPDFFile
- IsPDFObject
- IsProtected
- IsQuery
- IsSimpleValue
- IsSOAPRequest
- IsSpreadsheetFile
- IsSpreadsheetObject
- IsStruct
- IsUserInAnyRole
- IsUserInRole
- IsUserLoggedIn
- IsValid
- IsWDDX
- IsXML
- IsXmlAttribute
- IsXmlDoc
- IsXmlElem
- IsXmlNode
• IsXmlRoot
• JavaCast
• JSStringFormat
• isSafeHTML
• InvalidateOauthAccessstoken
• IsValidOauthAccessstoken
• InvokeCFClientFunction
**IncrementValue**

**Description**

Adds one to an integer.

**Returns**

The integer part of *number*, incremented by one.

**Category**

**Mathematical functions**

**Function syntax**

```
IncrementValue(number)
```

**See also**

**DecrementValue**

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Number to increment</td>
</tr>
</tbody>
</table>

**Example**

```html
<h3>IncrementValue Example</h3>
<p>Returns the integer part of a number incremented by one.</p>
<p>IncrementValue(0): <cfoutput>#IncrementValue(0)#</cfoutput></p>
<p>IncrementValue("1"): <cfoutput>#IncrementValue("1")#</cfoutput></p>
<p>IncrementValue(123.35): <cfoutput>#IncrementValue(123.35)#</cfoutput>
```
**InputBaseN**

**Description**

Converts *string*, using the base specified by *radix*, to an integer.

**Returns**

A number in the range 2-36, as a string.

**Category**

Mathematical functions

**Function syntax**

```
InputBaseN(string, radix)
```

**See also**

FormatBaseN

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String that represents a number, in the base specified by <em>radix</em>.</td>
</tr>
<tr>
<td>radix</td>
<td>Base of the number represented by <em>string</em>, in the range 2-36.</td>
</tr>
</tbody>
</table>

**Example**
<h3>InputBaseN Example</h3>

<p>FormatBaseN converts a number to a string in the base specified by Radix.</p>

<cfoutput>
<br>FormatBaseN(10,2): #FormatBaseN(10,2)#
<br>FormatBaseN(1024,16): #FormatBaseN(1024,16)#
<br>FormatBaseN(125,10): #FormatBaseN(125,10)#
<br>FormatBaseN(10.75,2): #FormatBaseN(10.75,2)#
</cfoutput>

<h3>InputBaseN Example</h3>

<p>InputBaseN returns the number obtained by converting a string, using the base specified by Radix.</p>

<cfoutput>
<br>InputBaseN("1010",2): #InputBaseN("1010",2)#
<br>InputBaseN("3ff",16): #InputBaseN("3ff",16)#
<br>InputBaseN("125",10): #InputBaseN("125",10)#
<br>InputBaseN(1010,2): #InputBaseN(1010,2)#
</cfoutput>
## Insert

**Description**

Inserts a substring in a string after a specified character position. If `position = 0`, prefixes the substring to the string.

**Returns**

A string.

**Category**

String functions

**Function syntax**

```
Insert(substring, string, position)
```

See also

- RemoveChars, Len

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>substring</code></td>
<td>A string or a variable that contains one. String to insert.</td>
</tr>
<tr>
<td><code>string</code></td>
<td>A string or a variable that contains one. String into which to insert <code>substring</code>.</td>
</tr>
<tr>
<td><code>position</code></td>
<td>Integer or variable; position in string after which to insert <code>substring</code>.</td>
</tr>
</tbody>
</table>

**Example**
<h3>Insert Example</h3>

<cfif IsDefined("FORM.myString")>
  <!--- if the position is longer than the length of the string, err --->
  <cfif FORM.insertPosition GT Len(MyString)>
    <cfoutput>
      <p>This string only has #Len(MyString)# characters; therefore, you cannot insert the substring #FORM.mySubString# at position #FORM.insertPosition#.
    </cfoutput>
  </cfif>
</cfif>

<cfelse>
  <cfoutput>
    <p>You inserted the substring #FORM.MySubstring# into the string #FORM.MyString#, resulting in the following string:<br>
    #Insert(FORM.MySubString, FORM.myString, FORM.insertposition)#
  </cfoutput>
</cfif>
**Int**

**Description**

Calculates the closest integer that is smaller than *number*. For example, it returns 3 for `Int(3.3)` and for `Int(3.7)`; it returns -4 for `Int(-3.3)` and for `Int(-3.7)`

**Returns**

An integer, as a string.

**Category**

Mathematical functions

**Function syntax**

```
Int(number)
```

**See also**

`Ceiling`, `Fix`, `Round`

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Real number to round down to an integer.</td>
</tr>
</tbody>
</table>

**Example**

```
<h3>Int Example</h3>
<p>Int returns the closest integer smaller than a number.</p>
<p>Int(11.7) : <cfoutput>#Int(11.7)#</cfoutput></p>
<p>Int(-11.7) : <cfoutput>#Int(-11.7)#</cfoutput></p>
<p>Int(0) : <cfoutput>#Int(0)#</cfoutput></p>
```
Invoke

Description

Does either of the following:

- Invokes a component method from within a ColdFusion page or component
- Invokes a web service

Returns

Returns what is returned by the invoked method.

History

ColdFusion 10: Added this function

Syntax

invoke(cfcinstance, methodname [, arguments])

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfcinstance</td>
<td>Required. String or component object; a reference to a component, or component to instantiate.</td>
</tr>
<tr>
<td>methodname</td>
<td>Required. Name of a method. For a web service, the name of an operation.</td>
</tr>
<tr>
<td>arguments</td>
<td>Optional. Arguments to pass to the method.</td>
</tr>
</tbody>
</table>

Example

```cfcscript
<cfscript>
  obj = createObject("component","TestComponent");
  invoke(obj,"function1",{a1="ColdFusion"});
</cfscript>
```
IsArray

Description

Determines whether a value is an array.

Returns

True, if value is an array, or a query column object.

Category

Array functions, Decision functions

Function syntax

IsArray(value [, number ])

See also

Array functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

History

ColdFusion MX:

- Changed behavior: if the value parameter contains a reference to a query result column, this function now returns True. For example: isArray(MyQuery Column1) returns True. (In earlier releases, it returns False.)
- Changed behavior: this function can be used on XML objects.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Variable or array name</td>
</tr>
<tr>
<td>number</td>
<td>Dimension; function tests whether the array has exactly this dimension</td>
</tr>
</tbody>
</table>

Usage

Use this function to determine whether a value is an array or query column. This function evaluates a Java array object, such as a vector object, as having one dimension.

Example
<h3>IsArray Example</h3>

<!--- Make an array --->
<cfset MyNewArray = ArrayNew(1)>

<!--- set some elements --->
<cfset MyNewArray[1] = "element one">
<cfset MyNewArray[2] = "element two">
<cfset MyNewArray[3] = "element three">

<!--- is it an array? --->
<cfoutput>
<p>Is this an array? #IsArray(MyNewArray)#</p>
<p>It has #ArrayLen(MyNewArray)# elements.</p>
<p>Contents: #ArrayToList(MyNewArray)#</p>
</cfoutput>
IsAuthenticated

Description

This function is obsolete. Use the newer security tools; see Conversion functions and Securing Applications in the Developing ColdFusion Applications.

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later ColdFusion releases.
IsAuthorized

Description

This function is obsolete. Use the newer security tools; see Conversion functions and Securing Applications in the Developing ColdFusion Applications

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later releases.
**IsBinary**

**Description**

Determines whether a value is stored as binary data.

**Returns**

True, if the value is binary; False, otherwise.

**Category**

Decision functions

**Function syntax**

```
IsBinary(value)
```

**See also**

ToBinary, ToBase64, IsNumeric, YesNoFormat

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Number or string</td>
</tr>
</tbody>
</table>

**Example**
<!--- Create a string of all ASCII characters (32-255) and concatenate them together. --->
<cfset charData ="">
<cfloop index="data" from="32" to="255">
    <cfset ch=chr(data)>
    <cfset charData=charData & ch>
</cfloop>

<b>The following string is the concatenation of all characters (32 to 255) from the ASCII table.</b><br><br>
<cfoutput>#htmleditformat(charData)#</cfoutput>
<br><br>
<!--- Create a Base 64 representation of this string. --->
<cfset data64=toBase64(charData)>

<!--- Convert string to binary. --->
<cfset binaryData=toBinary(data64)>

<!--- Check to see if it really is a binary value. --->
<cfif IsBinary(binaryData)>
The binaryData variable is binary!<br>
</cfif>

<!--- Convert binary data back to Base 64. --->
<cfset another64=toBase64(binaryData)>

<cfif Not IsBinary(another64)>
The another64 variable is NOT binary!<br></cfif>

<!--- Compare another64 with data64 to ensure that they are equal. --->
<cfif another64 eq data64>
Base 64 representation of binary data is identical to the Base 64 representation of string data.
<cfelse>
<h3>Conversion error.</h3>
</cfelse>
**IsBoolean**

**Description**

Determines whether a value can be converted to Boolean

**Returns**

True, if the parameter can be converted to Boolean; False, otherwise.

**Category**

Decision functions

**Function syntax**

```cfml
IsBoolean(value)
```

**See also**

IsNumeric, IsValid, YesNoFormat

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Number or string</td>
</tr>
</tbody>
</table>

**Example**

```cfml
<h3>IsBoolean Example</h3>

<cfif IsDefined("FORM.theTestValue")>
  <cfif IsBoolean(FORM.theTestValue)>
    <h3>The expression <cfoutput>#DE(FORM.theTestValue)#</cfoutput> is Boolean</h3>
  </cfif>
<cfelse>
  <h3>The expression <cfoutput>#DE(FORM.theTestValue)#</cfoutput> is not Boolean</h3>
</cfif>

<form action = "isBoolean.cfm">
  <p>Enter an expression, and discover whether it can be evaluated to a Boolean value.</p>
  <input type = "Text" name = "TheTestValue" value = "1">
  <input type = "Submit" value = "Is it Boolean?" name = ">
</form>
```
**IsClosure**

**Description**

Determines whether a value or expression references a function defined via a function expression, as opposed to a function statement.

**Returns**

True, if the value represents a function expression; otherwise False.

**Category**

Decision functions

**Syntax**

isClosure(value)

**See also**

Other decision functions.

**History**

ColdFusion 10: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A value.</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to determine whether the value or expression represents a function defined via a function expression.

**Example**

```cfs<br>`<br>/* a custom function defined via a function statement */<br>function multiplier(x,y){<br>  return x * y;<br>}</br>
<br>/* a custom function defined via a function expression */<br>adder = function(x,y){<br>  return x + y;<br>};<br>/* a string containing the name of the function reference */
name = "multiplier";<br></cfs`

```
<!-- examples using function references-->

<!---

a function declared via a function statement
IS a custom function,
but IS NOT a "closure"
--->

isCustomFunction(referenceToDeclaredFunction): #isCustomFunction(multiplier)#<br> <!--- true --->

isClosure(referenceToDeclaredFunction): #isClosure(multiplier)#<br><hr> <!--- false --->

<!---

a function declared via a function expression
IS NOT a custom function,
but IS a "closure"
--->

isCustomFunction(referenceToFunctionExpression): #isCustomFunction(adder)#<br> <!--- false --->

isClosure(referenceToFunctionExpression): #isClosure(adder)#<br><hr> <!--- true --->

<!--- examples not using function references-->

<!--- a string IS NOT a custom function, nor is it a closure --->

isCustomFunction(stringContainingNameOfFunctionReference): #isCustomFunction(name)#<br> <!--- false --->

isClosure(stringContainingNameOfFunctionReference): #isClosure(name)#<br><hr> <!--- false --->

<!---

a string-literal containing the function expression variable name
is not correct usage: it’s just a string
--->

isCustomFunction("adder"): #isCustomFunction("adder")#<br> <!--- false --->

isClosure("adder"): #isClosure("adder")#<br><hr> <!--- false --->
</cfoutput>
<cfscript>
// an expression can be evaluated directly
result = isCustomFunction(function(x,y){
  return x-y;
})
);

writeOutput("isCustomFunction(lineExpression): #result#<br>"); // false

result = isClosure(function(x,y){
  return x-y;
});
</cfscript>
writeOutput("isClosure(inlineExpression): #result#<br>"); // true
</cfscript>
**IsCustomFunction**

**Description**

Determines whether a name represents a custom function.

**Returns**

True, if `name` can be called as a custom function; False, otherwise.

**Category**

Decision functions

**Function syntax**

```plaintext
IsCustomFunction(name)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of a custom function. Must not be in quotation marks. If not a defined variable or function name, ColdFusion generates an error.</td>
</tr>
</tbody>
</table>

**Usage**

The `IsCustomFunction` function returns True for any function that can be called as a custom function, including functions defined using CFScript `function` declarations and `cffunction` tags, and functions that are ColdFusion component methods. For CFC methods, first instantiate the component.

**Note**

To prevent undefined variable exceptions, always precede `IsCustomFunction` with an `IsDef` test, as shown in the example.

**Note**

This function does not return true if the function was defined using a function expression, eg:

```javascript
realUDF = function() {
    return 1;
}
```

For functions defined using function expressions, use `isClosure()`.

**Example**
<h3>IsCustomFunction Example</h3>
<cfscript>
function realUDF() {
    return 1;
}
</cfscript>
<cfset X = 1>

<!--- Example that fails existence test --->
<cfif IsDefined("Foo") AND IsCustomFunction(Foo)>
    Foo is a UDF.<br>
</cfif>

<!--- Example that passes existence test but fails IsCustomFunction --->
<cfif IsDefined("X") AND IsCustomFunction(X)>
    X is a UDF.<br>
</cfif>

<!--- Example that passes both tests--->
<cfif IsDefined("realUDF") AND IsCustomFunction(realUDF)>
    realUDF is a function.<br>
</cfif>

<!--- Example using a CFC, defined in TestCFC.cfc--->
<cfobject component="TestCFC" name="myTestCFCObject">
<CFIF IsDefined("myTestCFCObject.testFunc") AND
    IsCustomFunction(myTestCFCObject.testFunc)>
    myTestCFCObject.testFunc is a function.
</CFIF>
IsDate

Description

Determines whether a string or Java object can be converted to a date/time value.

Returns

True, if string can be converted to a date/time value; False, otherwise. ColdFusion converts the Boolean return value to its string equivalent, "Yes" or "No."

Category

Date and time functions, Decision functions

Function syntax

IsDate(string)

See also

CreateDateTime, IsNumericDate, IsValid, LSDateFormat, LSIsDate, ParseDateTime

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one.</td>
</tr>
</tbody>
</table>

Usage

This function checks against U.S. date formats only. For other date support, see LSDateFormat. A date/time object falls in the range 100 AD-9999 AD.

Example
<h3>IsDate Example</h3>
<cfif IsDefined("FORM.theTestValue")>
    <cfif IsDate(FORM.theTestValue)>
        <h3>The string <cfoutput>#DE(FORM.theTestValue)#</cfoutput>
            is a valid date</h3>
    </cfif>
    <cfelse>
        <h3>The string <cfoutput>#DE(FORM.theTestValue)#</cfoutput>
            is not a valid date</h3>
    </cfelse>
</cfif>
</cfif>
<form action = "isDate.cfm" method="post">
    <p>Enter a string, find whether it can be evaluated to a date value.</p>
    <p><input type = "Text" name = "TheTestValue" value = "&lt;cfoutput>#Now()#"
        &lt;/cfoutput>"
    <input type = "Submit" value = "Is it a Date?" name = "">
</form>
IsDDX

Description

Determines whether a DDX file exists and is valid, or if a string contains valid DDX instructions.

Returns

True, if the value represents a valid DDX file or string. False, otherwise.

Category

Decision functions

Function syntax

IsDDX("path or string")

See also

IsPDFObject, IsPDFFile, cfpdf

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path or string</td>
<td>Pathname to the DDX file or a string of DDX instructions. The pathname can be absolute or relative to the CFM page that calls it and must be enclosed in quotation marks.</td>
</tr>
</tbody>
</table>

Usage

This function returns False if the pathname to the DDX file is invalid, the pathname to the DDX file is null, the DDX file does not conform to the schema supported by ColdFusion, or the DDX instructions are invalid.

Example
<cfif IsDDX("TOCformat.ddx")>
  <cfset inputStruct=StructNew()>
  <cfset inputStruct.Doc0="title.pdf"/>
  <cfset inputStruct.Doc1="Chap1.pdf"/>
  <cfset inputStruct.Doc2="Chap2.pdf"/>
  <cfset inputStruct.Doc3="Chap3.pdf"/>
  <cfset inputStruct.Doc4="Chap4.pdf"/>

  <cfset outputStruct=StructNew()>
  <cfset outputStruct.Out1="myBook.pdf"/>

  <cfpdf action="processddx" ddxfile="TOCformat.ddx" inputfiles="#inputStruct#" outputfiles="#outputStruct#" name="ddxVar">
    <cfoutput>#ddxVar.Out1#</cfoutput>
  </cfpdf>
</cfif>

<cfelse>
  <p>This is not a valid DDX file.</p>
</cfelse>
IsDebugMode

Description

Determines whether debugging output is enabled.

Returns

True, if debugging mode is set in the ColdFusion Administrator; False if debugging mode is disabled.

Category

Decision functions

Function syntax

IsDebugMode()

See also

cfsetting

Description

If debugging output is enabled in ColdFusion Administrator and has not been overridden by setting the cfsetting tag showDebugOutput attribute to No, the IsDebugMode function returns Yes; No, otherwise.

Example

<h3>IsDebugMode Example</h3>  
<cfif IsDebugMode()>
  <h3>Debugging is set in the ColdFusion Administrator</h3>
<cfelse>
  <h3>Debugging is disabled</h3>
</cfif>
IsDefined

Description

Evaluates a string value to determine whether the variable named in it exists. This function is an alternative to the `ParameterExists` function, which is deprecated.

Returns

True, if the variable is found; False, otherwise.

Category

Decision functions

Function syntax

```
IsDefined("variable_name")
```

See also

Evaluate

History

ColdFusion MX: Changed behavior: this function can process only the following constructs:

- A simple variable
- A named variable with dot notation
- A named structure with dot notation (for example: `mystruct.key`)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable_name</td>
<td>String, enclosed in quotation marks. Name of variable to test for.</td>
</tr>
</tbody>
</table>

Usage

Passing an array entry, such as `myArray3` to this function causes an error. To check whether a specific entry exists in an array, use the `ArrayIsDefined` function. You can test whether a specific key exists in a structure by using this function or the `StructKeyExists` function. For example, when working with scopes that ColdFusion exposes as structures, the `StructKeyExists` function can sometimes replace this function. The following lines are equivalent:

```
if(isDefined("form.myVariable"))
if(structKeyExists(form,"myVariable"))
```

Example
<cfif IsDefined("form.myString")>
  <p>The variable form.myString has been defined, so show its contents. This construction allows us to place a form and its resulting action code on the same page and use IsDefined to control the flow of execution.</p>
  <p>The value of "form.myString" is <b><i><cfoutput>#form.myString#</cfoutput></i></b></p>
<cfelse>
  <p>During the first time through this template, the variable "form.myString" has not yet been defined, so we do not try to evaluate it.</p>
</cfif>

<form action="#CGI.Script_Name" method="POST">
<input type="Text" name="MyString" value="My sample value">
<input type="Submit" name=""/>
</form>
**IsImage**

**Description**

Determines whether a variable returns a ColdFusion image.

**Returns**

True, if the value is a ColdFusion image; False, otherwise.

**Category**

*Image functions*

**Function syntax**

```
IsImage(name)
```

**See also**

cfimage, ImageGetBlob, ImageInfo, ImageNew

**History**

ColdFusion 8: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required. The ColdFusion variable that is checked.</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to determine whether a variable returns a ColdFusion image.

**Example**

```cfc
cif IsImageFile("images/#form.art#")>
cfset myImage=ImageNew("images/#form.art#")>
...
cfset IsImage("#myImage#")>
cfimage action="writeToBrowser" source="#myImage#">
</cif>
```
**IsImageFile**

Description

Verifies whether an image file is valid.

Returns

True, if the value is a valid image file; False, otherwise.

Category

Image functions

Function syntax

```
IsImageFile("path")
```

See also

cfimage, ImageGetBlob, ImageInfo, ImageNew, IsImage

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Required. The pathname of the on-disk or in-memory file, or URL, to be checked. The pathname can be absolute or relative to the CFM page and must be enclosed in quotation marks.</td>
</tr>
</tbody>
</table>

Usage

Use this function to determine whether an image file is valid. This function returns a False value if the image file format is not supported by the server where ColdFusion is deployed, or if the pathname to the image file is null or invalid. For a list of standard image file formats supported by ColdFusion, see the supported image file formats provided in cfimage. To determine which image file formats are supported on the server where ColdFusion is deployed, use the GetReadableImageFormats and GetWriteableImageFormats.

Example
<cfif IsImageFile("images/#artwork.largeImage#")>
  <cfset myImage=ImageNew("images/#artwork.largeImage#")>
  <cfset ImageResize(myImage,50,"")>
  <cfimage action="writeToBrowser" source="#myImage#"/>
<cfelse>
  <p>I'm sorry, there is no image associated with the title you selected. Please click the Back button and try again.</p>
</cfif>
IsInstanceOf

Description

Determines whether an object is an instance of a ColdFusion interface or component, or of a Java class.

Returns

Returns true if any of the following is true:

- The object specified by the first parameter is an instance of the interface or component specified by the second parameter.
- The Java object specified by the first parameter was created by using the `cfobject` tag or `CreateObject` method and is an instance of the Java class specified by the second parameter.
- The object specified by the first parameter is an instance of a component that extends the component specified in the second parameter.
- The object specified by the first parameter is an instance of a component that extends a component that implements the interface specified in the second parameter.
- The Java object specified by the first parameter was created by using the `cfobject` tag or `CreateObject` method and is an instance of a Java class that extends the class specified by the second parameter. Returns false otherwise.

Note

The `IsInstanceOf` function returns `false` if the CFC specified by the `object` parameter does not define any functions.

Category

Decision functions, Extensibility functions

Function syntax

```
IsInstanceOf(object, typeName)
```

See also

cfcomponent, cfinterface, cfobject

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The CFC instance or Java object that you are testing</td>
</tr>
<tr>
<td>typeName</td>
<td>The name of the interface, component, or Java class of which the object might be an instance</td>
</tr>
</tbody>
</table>
Usage

For Java objects, the comparison is valid only if you created the object specified in the first parameter by using a cfobject tag or CreateObject method.

Example

<!--- to use this example, create an I1.cfc interface, as follows: --->
<cfinterface>
  <cffunction name = "method1">
  </cffunction>
</cfinterface>

<!--- Create a C1.cfc component that implements the I1.cfc interface, as follows: --->
<cfcomponent implements=I1>
  <cffunction name = "method1">
    <cfoutput>C1.method1 called</cfoutput>
  </cffunction>
</cfcomponent>

<!--- Create a test.cfm file as follows and display the page. --->
<!--- Create an instance of the C1 CFC, which implements the I1 interface. --->
<cfset c1obj = CreateObject("Component", "C1")>

<!--- Create a Java object --->
<cfset javaObj = createobject("java", "java.lang.System")>

<cfoutput>
  IsInstanceOf(c1obj,"C1") = #IsInstanceOf(c1obj,"C1")# (Expected = YES)<br>
  IsInstanceOf(c1obj,"I1") = #IsInstanceOf(c1obj,"I1")# (Expected = YES)<br>
  IsInstanceOf(c1obj,"C2") = #IsInstanceOf(c1obj,"C2")# (Expected = NO)<br>
  IsInstanceOf(javaObj,"java.lang.System") =
    #IsInstanceOf(javaObj,"java.lang.System")# (Expected = YES)<br>
  IsInstanceOf(javaObj,"java.lang.String") =
    #IsInstanceOf(javaObj,"java.lang.String")# (Expected = NO)<br>
</cfoutput>
IsIPv6

Description

Determines whether the host supports IPv6.

Returns

True if the host supports IPv6.

Category

Decision functions

Syntax

IsIPv6()
IsIPv6(hostname)

See also

GetLocalHostIP, IsLocalHost

History

ColdFusion 9: Added this function

Usage

When you use this function to verify if the remote host supports IPv6, pass the hostname and not the IP address. This function applies only if the server that places the request is IPv6-enabled.

Example

The following example checks whether the localhost supports IPv6:

```
<cfset hostname="localhost">
<cfset test=IsIPv6()>
<cfif test>
<cfoutput>localhost supports IPv6</cfoutput>
</cfif>
```

The following example checks whether the remote host supports IPv6:

```
<cfset hostname="remote hostname">
<cfset test=IsIPv6(hostname)>
<cfif test>
<cfoutput>#hostname# supports IPv6</cfoutput>
</cfif>
```
IsJSON

Description

Evaluates whether a string is in valid JSON (JavaScript Object Notation) data interchange format.

Returns

True if the parameter is a valid JSON value. False if the parameter is not a valid JSON data representation.

Category

Conversion functions

Syntax

IsJSON(var)

See also

DeserializeJSON, SerializeJSON, cfajaxproxy, Using Ajax Data and Development Features in the Developing ColdFusion Applications, http://www.json.org

History

ColdFusion 8: Added function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>A string or variable that represents one.</td>
</tr>
</tbody>
</table>

Example

This example checks whether the data feed that is generated by the example for the SerializeJSON function contains valid JSON data. The feed is in the form of a JavaScript function call where the parameter is a JSON string that contains the feed data. The example does the following operations:

1. Uses a cfhttp tag to get the feed (in the cfhttp.fileContent variable).
2. Strips the function call wrapper from the text.
3. Uses the IsJSON function to check whether the result of the previous step is a valid JSON format string.
4. Displays a message that indicates whether the text is valid JSON data, followed by the feed text string. To run this example, put this file and the example for the SerializeJSON function in an appropriate location under your ColdFusion web root, replace the URL with the correct URL for the serialization example, and run this page.
<!--- Get the JSON Feed --->
<cfhttp url="http://localhost:8500/My_Stuff/Ajax/Books/CreateJSON_NEW.cfm">
<!--- JSON data is often distributed as a JavaScript function. The following REReplace functions strip the function wrapper. --->
<cfset theData=REReplace(cfhttp.FileContent,
   "^\s*\[[[:word:]]]*\s*\("","")>
<cfset theData=REReplace(theData, "\s*\)\s*$", "" )>

<!--- Test to make sure you have JSON data. --->
<cfif isJSON(theData)>
   <h3>The URL you requested provides valid JSON</h3>
<cfelse>
   <h3>The URL you requested does not provide valid JSON</h3>
</cfif>

<!--- Display the data. --->
<cfoutput>#theData#</cfoutput>

For a more complex example that then converts the JSON data, see DeserializeJSON.
IsK2ServerABroker

Description

This function is deprecated.

Returns

True, if K2Broker is in configured with ColdFusion; False, otherwise.

Category

Decision functions, Full-text search functions, Query functions

Function syntax

IsK2ServerABroker()

See also


History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases.ColdFusion MX: Added this function.

Example

<cfoutput>IsK2ServerABroker = $*#IsK2ServerABroker()#*$</cfoutput>
IsK2ServerDocCountExceeded

Description

This function is deprecated.

Returns

True, if the document count limit is exceeded; False, otherwise.

Category

Decision functions, Full-text search functions, Query functions

Function syntax

IsK2ServerDocCountExceeded()

See also

GetK2ServerDocCountLimit, IsK2ServerABroker

History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases.
ColdFusion 5: Added this function.

Example

IsK2ServerOnline

Description

This function is deprecated because the K2Server is always running when ColdFusion is running.

Returns

True, if the K2Server is available to perform a search; False, otherwise.

Category

Decision functions, Full-text search functions, Query functions

Function syntax

```
IsK2ServerOnline()
```

See also

IsK2ServerABroker

History

ColdFusion MX 6.1: Deprecated this function. It might not work, and it might cause an error, in later releases.

ColdFusion MX: Added this function.

Example

```
<cfoutput>IsK2ServerOnline = 
    $*#IsK2ServerOnline()#*$</cfoutput>
```
IsLeapYear

Description

Determines whether a year is a leap year.

Returns

True, if year is a leap year; False, otherwise.

Category

Date and time functions, Decision functions

Function syntax

IsLeapYear(year)

See also

DaysInYear

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Number representing a year</td>
</tr>
</tbody>
</table>

Example

```
<h3>IsLeapYear Example</h3>

<cfif IsDefined("FORM.theTestValue")>
    <cfif IsLeapYear(FORM.theTestValue)>
        <h3>The year value `<cfoutput>#DE(FORM.theTestValue)#</cfoutput>` is a Leap Year</h3>
    </cfif>
    <cfelse>
        <h3>The year value `<cfoutput>#DE(FORM.theTestValue)#</cfoutput>` is not a Leap Year</h3>
    </cfelse>
</cfif>

<form action="isLeapYear.cfm" method="POST">
    <p>Enter a year value, and find out whether it is a Leap Year.</p>
    <p><input type="Text" name="TheTestValue" value="#Year(Now())#"></p>
    <input type="Submit" value="Is it a Leap Year?" name=""></form>
```
IsLocalHost

Description

Determines whether the specified IP address is the localhost. This supports both IPv4 and IPv6 addresses.

Returns

True, if the IP address is the localhost; False, otherwise.

Category

Decision functions

Function syntax

IsLocalHost(ipaddress)

See also

GetLocalHostIP

History

ColdFusion MX 7.01: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipaddress</td>
<td>Valid IP address.</td>
</tr>
</tbody>
</table>

Example

```cfml
<h3>IsLocalHost Example</h3>

<cfif IsDefined("FORM.theTestIPAddress")>
  <cfif IsLocalHost(FORM.theTestIPAddress)>
    <h3>The IP address <cfoutput>#FORM.theTestIPAddress#</cfoutput> is the localhost.</h3>
  </cfif>
  <cfelse>
    <h3>The IP address <cfoutput>#DE(FORM.theTestIPAddress)#</cfoutput> is not the localhost.</h3>
  </cfif>
</cfif>

<form action = "isIPAddressLocalHost.cfm">
  <p>Enter an IP address to find out if it is the localhost.</p>
  <p><input type = "Text" name = "TheTestIPAddress" value = "127.0.0.1" /></p>
</form>
```
IsNull

Description

Used to check if the given object or expression evaluates to null.

Returns

True, if the given object is null or if the given expression evaluates to null; False, otherwise.

Category

Decision functions

Function syntax

IsNull(obj)

History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>Object for which you perform the null check.</td>
</tr>
</tbody>
</table>
IsNumeric

Description

Determines whether a string can be converted to a numeric value. Supports numbers in U.S. number format. For other number support, use `LSIsNumeric`.

Returns

True, if `string` can be converted to a number; False, otherwise.

Category

Decision functions

Function syntax

```
IsNumeric(string)
```

See also

`IsBinary`, `IsValid`

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one.</td>
</tr>
</tbody>
</table>

Example

```html
<h3>IsNumeric Example</h3>
<cfif IsDefined("FORM.theTestValue")>
  <cfif IsNumeric(FORM.theTestValue)>
    <h3>The string `<cfoutput>#DE(FORM.theTestValue)#</cfoutput>` can be converted to a number</h3>
  </cfif>
</cfif>

<cfelse>
  <h3>The string `<cfoutput>#DE(FORM.theTestValue)#</cfoutput>` cannot be converted to a number</h3>
</cfif>

<form action = "isNumeric.cfm">
  <p>Enter a string, and find out whether it can be evaluated to a numeric value.</p>
  <p><input type = "Text" name = "TheTestValue" value = "123"> <input type = "Submit" value = "Is it a Number?" name = ">
</form>
```
IsNumericDate

Description

Evaluates whether a real number is a valid representation of a date (date/time object).

Returns

True, if the parameter represents a valid date/time object; False, otherwise.

Category

Date and time functions, Decision functions

Function syntax

IsNumericDate(number)

See also

IsDate, ParseDateTime

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A real number</td>
</tr>
</tbody>
</table>

Usage

ColdFusion, by default, evaluates any input parameter and attempts to convert it to a real number before it passes the parameter to the IsNumericDate function. As a result, parameter values such as 12/12/03 and `{ts '2003-01-14 10:04:13'}` return True, because ColdFusion converts valid date string formats to date/time objects, which are real numbers.

Example
<h3>IsNumericDate Example</h3>
<cfif IsDefined("form.theTestValue")>
<!--- test if the value represents a number or a pre-formatted Date value --->
<cfif IsNumeric(form.theTestValue) or IsDate(form.theTestValue)>
<!--- if this value is a numericDate value, then pass --->
<cfif IsNumericDate(form.theTestValue)>
<h3>The string <cfoutput>#DE(form.theTestValue)#</cfoutput> can be converted to a valid numeric date</h3>
</cfif>
<h3>The string <cfoutput>#DE(form.theTestValue)#</cfoutput> can not be converted to a valid numeric date</h3>
</cfif>
<cfelse>
<h3>The string <cfoutput>#DE(form.theTestValue)#</cfoutput> is not a valid numeric date</h3>
</cfif>
</cfif>

<form action="#cgi.script_name#" method="POST">
<p>Enter a value, and discover if it can be evaluated to a date value.
<p>
<input type="Text" name="TheTestValue" value="<CFOUTPUT>#Now()#</CFOUTPUT>">
<input type="Submit" value="Is it a Date?" name="">
</form>
**IsObject**

**Description**

Determines whether a value is an object.

**Returns**

True, if the value represents a ColdFusion object. False if the value is any other type of data, such as an integer, string, date, or struct.

**Category**

**Decision functions**

**Function syntax**

```
IsObject(value)
```

**See also**

IsDate, IsImage, IsNumeric, IsNumericDate, IsQuery, IsSimpleValue, IsStruct, IsWDDX, IsXmlDoc, IsXmlElem, IsXmlRoot

**History**

ColdFusion MX: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A value, typically the name of a variable.</td>
</tr>
</tbody>
</table>

**Usage**

This function returns False for query and XML objects.

**Example**
<!--- to use this example, create a color.cfc component as follows: --->
<!---
<cfcomponent>
<cffunction name="myFunction" access="public" returntype="string">
<!--- Create a structure object --->
<cfset myColor = "Blue">
<cffunction>
</cfcomponent>
<cfobject name="getColor" component="color">

<cfif IsObject(getColor)>
<!--- Invoke the myFunction method --->
<cfinvoke
    component="#getColor#"
    method="myFunction"
    returnVariable="myColor">
</cfinvoke>

<cfif IsDefined("myColor")>
<!--- Output the returned variable --->
The value of myColor = <cfoutput>#myColor#</cfoutput><p>
</cfif>
</cfif>
</cfobject>
isOnline

Description

Determines whether the gateway is connected to the instant messaging server.

Syntax

Boolean = isOnline()

See also

getCustomAwayMessage, getStatusAsString, getStatusTimeStamp, setStatus, Using the GatewayHelper object in the Developing ColdFusion Applications

Returns

True, if the gateway is connected to the IM server; False, otherwise.

Example

See GatewayHelper example, in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.----

#back to top
IsPDFFile

Description

Verifies whether a PDF file is valid.

Returns

True, if the value returns a valid PDF file. False, otherwise.

Category

Decision functions

Function syntax

IsPDFFile("path")

See also


History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>Pathname to an on-disk or in-memory PDF file. The pathname can be absolute or relative to the CFM page and must be enclosed in quotation marks.</td>
</tr>
</tbody>
</table>

Usage

This function returns False if the value is not a valid pathname to a PDF file, the pathname is null, the PDF file is not valid, or the PDF file is corrupted.

Example

```cftm
<!--- The following code shows the action page for a form where a user chooses a PDF document to print. --->

<cfif IsPDFFile("#form.printMe#")>
  <cfprint type="PDF" source="#myPDF#">
<cfelse>
  <p>This is not a valid PDF file or the PDF document you have chosen is not available.</p>
</cfif>
```
IsPDFObject

Description
Determines whether a value is a PDF object.

Returns
True, if the value represents a PDF object. False if the value is any other type of data, such as an integer, string, date, or structure.

Category
Decision functions

Function syntax

IsPDFObject(value)

See also

History
ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A value, typically the PDF object stored as a variable name.</td>
</tr>
</tbody>
</table>

Usage
This function returns False for query and XML objects.

Example

```<cfpdf source="c:\forms\uoteform.pdf" action="read" name="myPDFform"/><cfif IsPDFObject(myPDFform)><cfpdf source=#myPDFform# action="write" destination = "c:\forms\newPDFForm.pdf"><cfelse><p>This is not a PDF.</p></cfif>```
IsProtected

Description

This function is obsolete. Use the newer security tools; see Conversion functions and Securing Applications in the Developing ColdFusion Applications

History

ColdFusion MX: This function is obsolete. It does not work in ColdFusion MX and later releases.
**IsQuery**

**Description**

Determines whether `value` is a query.

**Returns**

True, if `value` is a query.

**Category**

Decision functions, Query functions

**Function syntax**

```coldfusion
IsQuery(value)
```

**See also**

[QueryAddRow](#)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Query variable</td>
</tr>
</tbody>
</table>

**Example**
<!--- Shows an example of IsQuery and IsSimpleValue --->
<h3>IsQuery Example</h3>
<!--- define a variable called "GetEmployees" --->
<CFPARAM name = "GetEmployees" DEFAULT = "#Now()#"

<p>Before the query is run, the value of GetEmployees is
<cfoutput>#GetEmployees#</cfoutput>

<cfif IsSimpleValue(GetEmployees)>
<p>GetEmployees is currently a simple value
</cfif>
<!--- make a query on the snippets datasource --->
<cfquery name = "GetEmployees" datasource = "cfdocexamples">
SELECT *
FROM employees
</cfquery>

<p>After the query is run, GetEmployees contains a number of rows
that look like this (display limited to three rows):
<cfoutput QUERY = "GetEmployees" MaxRows = "3">
<pre>#Emp_ID# #FirstName# #LastName#</pre>
</cfoutput>
<cfif IsQuery(GetEmployees)>
GetEmployees is no longer a simple value, but the name of a query
</cfif>
**IsSimpleValue**

**Description**

Determines the type of a value.

**Returns**

True, if value is a string, number, Boolean, or date/time value; False, otherwise.

**Category**

**Decision functions**

**Function syntax**

```
IsSimpleValue(value)
```

**See also**

**IsValid**

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Variable or expression</td>
</tr>
</tbody>
</table>

**Example**
<h3>IsSimpleValue Example</h3>

<!--- define a variable called "GetEmployees" --->
<cfparam name = "GetEmployees" default = "#Now()#">

<p>Before the query is run, the value of GetEmployees is</p>
<cfoutput>#GetEmployees#</cfoutput>

<cfif IsSimpleValue(GetEmployees)>
<p>GetEmployees is currently a simple value</p>
</cfif>

<!--- make a query on the snippets datasource --->
<cfquery name = "GetEmployees" datasource = "cfdocexamples">
SELECT *
FROM employees
</cfquery>

<p>After the query is run, GetEmployees contains a number of rows that look like this (display limited to three rows):</p>
<cfoutput QUERY = "GetEmployees" MaxRows = "3">
<pre>#Emp_ID# #FirstName# #LastName#</pre>
</cfoutput>

<cfif IsQuery(GetEmployees)>
GetEmployees is no longer a simple value, but the name of a query
</cfif>
IsSOAPRequest

Description

Determines whether a CFC is being called as a web service.

Returns

True if CFC is being called as a web service; False, otherwise.

Category

XML functions

History

ColdFusion MX 7: Added this function.

Function syntax

IsSOAPRequest()

See also

AddSOAPRequestHeader, AddSOAPResponseHeader, GetSOAPRequest, GetSOAPRequestHeader, GetSOAPResponse, GetSOAPResponseHeader; Basic web service concepts in Web services in the Developing ColdFusion Applications

Usage

Call this function within a CFC to determine if it is running as a web service.

Example

This example creates a CFC web service that illustrates the operation of the IsSOAPRequest function and also provides a web service that illustrates the operation of other ColdFusion SOAP functions. Save the following code as headerservice.cfc in a folder called soapheaders under your web root. Test its operation, and specifically the operation of the IsSOAPRequest function, by executing the examples that invoke this web service. For example, see the example for AddSOAPRequestHeader.
<h3>IsSOAPRequest Example</h3>
<cfcomponent displayname="tester" hint="Test for SOAP headers">

<cffunction name="echo_me"
access="remote"
output="false"
returntype="string"
displayname="Echo Test" hint="Header test">
<cfargument name="in_here" required="true" type="string">

<cfset isSOAP = isSOAPRequest()>
<cfif isSOAP>

<!--- Get the first header as a string and as XML --->
<cfset username = getSOAPRequestHeader("http://mynamespace/", "username")>
<cfset return = "The service saw username: " & username>
<cfset xmlusername = getSOAPRequestHeader("http://mynamespace/", "username", "TRUE")>
<cfset return = return & "<br> as XML: " & xmlusername>

<!--- Get the second header as a string and as XML --->
<cfset password = getSOAPRequestHeader("http://mynamespace/", "password")>
<cfset return = return & "The service saw password: " & password>
<cfset xmlpassword = getSOAPRequestHeader("http://mynamespace/", "password", "TRUE")>
<cfset return = return & "<br> as XML: " & xmlpassword>

<!--- Add a header as a string --->
<cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED VALUE", false)>

<!--- Add a second header using a CFML XML value --->
<cfset doc = XmlNew()>
<cfset x = XmlElemNew(doc, "http://www.tomj.org/myns", "returnheader2")>
<cfset x.XmlText = "hey man, here I am in XML">
<cfset x.XmlAttributes["xsi:type"] = "xsd:string">
<cfset tmp = addSOAPResponseHeader("ignoredNameSpace", "ignoredName", x)>

<cfelse>

<!--- Add a header as a string - Must generate error! --->
<cfset addSOAPResponseHeader("http://www.tomj.org/myns", "returnheader", "AUTHORIZED VALUE", false)>

<cfset return = "Not invoked as a web service">
</cfif>

<cfreturn return>
</cffunction>
</cfcomponent>
IsSpreadsheetFile

Description

Returns a value that determines if the input is a spreadsheet file.

Returns

A Boolean value. True if the input is a spreadsheet file; False if it is a spreadsheet object.

Category

Microsoft Office Integration

Function syntax

IsSpreadsheetFile(spreadsheetfile)

See also

History

ColdFusion 9: Added.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetfile</td>
<td>Specifies the spreadsheet file.</td>
</tr>
</tbody>
</table>

Example

```
<cfset filepath = "C:/Documents/SingleSheet.xls">
<cfoutput># isSpreadSheetObject(filepath)# NOT an Object<br></cfoutput>
<cfoutput># isSpreadSheetFile(filepath)# a File<br></cfoutput>
```
IsSpreadsheetObject

Description

Returns a value that determines if the input is a spreadsheet object.

Returns

A Boolean value. True if the input is a spreadsheet object; False if it is a spreadsheet file.

Category

Microsoft Office Integration

Function syntax

IsSpreadsheetObject(spreadsheetobject)

See also

IsSpreadsheetFile

History

ColdFusion 9: Added.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobject</td>
<td>Specifies the spreadsheet object.</td>
</tr>
</tbody>
</table>

Example

```
<cfspreadsheet action="read" src="C:/documents/SingleSheet.xls"
name="SpreadsheetObj" >
<cfoutput># isSpreadSheetObject(SpreadsheetObj)# an Object<br></cfoutput>
<cfoutput># isSpreadSheetFile(SpreadsheetObj)# NOT a File<br></cfoutput>
```
IsStruct

Description

Determines whether a variable is a structure.

Returns

True, if variable is a ColdFusion structure or is a Java object that implements the java.lang.Map interface. The return value is False if the object in variable is a user-defined function (UDF).

Category

Decision functions, Structure functions

Function syntax

IsStruct(variable)

See also

Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

History

ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable</td>
<td>Variable name</td>
</tr>
</tbody>
</table>

Example
<!--- This view-only example shows the use of IsStruct. --->
<p>This file is similar to addemployee.cfm, which is called by StructNew, StructClear, and StructDelete. It is an example of a custom tag used to add employees. Employee information is passed through the employee structure (the EMPINFO attribute). In UNIX, you must also add the Emp_ID.
<!---
<cfswitch expression = "#ThisTag.ExecutionMode#">
<cfcase value = "start">
<cfif IsStruct(attributes.EMPINFO)>
<cfoutput>Error. Invalid data.</cfoutput>
<cfexit method = "ExitTag">
<cfelse>
<cfquery name = "AddEmployee" datasource = "cfdocexamples">
INSERT INTO Employees
(FirstName, LastName, Email, Phone, Department)
VALUES
<cfoutput>
{
'#{StructFind(attributes.EMPINFO, "firstname")#}',
'#{StructFind(attributes.EMPINFO, "lastname")#}',
'#{StructFind(attributes.EMPINFO, "email")#}',
'#{StructFind(attributes.EMPINFO, "phone")#}',
'#{StructFind(attributes.EMPINFO, "department")#}'
}
</cfoutput>
</cfquery>
</cfif>
</cfcase>
</cfswitch> --->
IsUserInAnyRole

Description

Determines whether an authenticated user belongs to any role in a list of roles.

Returns

True, if the authenticated user, belongs to any Role in the list; False, otherwise.

Category

Security functions, Decision functions

Function syntax

```
IsUserInAnyRole(role_list)
```

See also

cflogin, cfloginuser, cfslogout, GetAuthUser, GetUserRoles, IsUserRole, IsUserLoggedIn, Securing Applications in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role_list</td>
<td>A comma-delimited list of one or more roles to be tested.</td>
</tr>
</tbody>
</table>

Example

```
<cfif IsUserInAnyRole(allRoles) >
  <cfoutput>Authenticated user is in these roles: #GetUserRoles()#</cfoutput>
<cfelseif >
  <cfoutput>Authenticated user is in no roles</cfoutput>
</cfif>
```
**IsUserInRole**

**Description**

Determines whether an authenticated user belongs to the specified Role.

**Returns**

True, if the authenticated user, belongs to the specified Role; False, otherwise.

**Category**

Security functions, Decision functions

**Function syntax**

```plaintext
IsUserInRole("role_name")
```

**See also**

cflogin, cfloginuser, GetAuthUser, GetUserRoles, IsUserInAnyRole, IsUserLoggedIn, Securing Applications in the Developing ColdFusion Applications

**History**

ColdFusion MX: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role_name</td>
<td>Name of a security role</td>
</tr>
</tbody>
</table>

**Usage**

Role names are not case-sensitive. To check if a user is in multiple roles, specify them in a comma-delimited list, such as "Admin,HR". Lists with multiple roles cannot contain spaces as separators; for example, do not use "Admin, HR".

**Example**

```plaintext
<cfif IsUserInRole("Admin")>
    <cfoutput>Authenticated user is an administrator</cfoutput>
<cfelse IsUserInRole("User")>
    <cfoutput>Authenticated user is a user</cfoutput>
</cfif>
```
IsUserLoggedIn

Description

Determines whether a user is logged in.

Returns

True, if the user, is logged in; False, otherwise.

Category

Security functions, Decision functions

Function syntax

IsUserLoggedIn()

See also

cflogin, cfloginuser, GetAuthUser, GetUserRoles, IsUserInAnyRole, IsUserInRole, Securing Applications in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function.

Example

<cfif IsUserLoggedIn()>
  <cfinclude template="welcome.cfm">
<cfelse>
  <cfinclude template="loginform.cfm">
  <cfabort>
</cfif>
IsValid

Description

Tests whether a value meets a validation or data type rule.

Returns

True, if the value conforms to the rule; False, otherwise.

Category

Decision functions

Function syntax

```
IsValid(type, value)
isValid("range", value, min, max)
isValid("regex" or "regular_expression", value, pattern)
```

See also

cfparam, cfform, IsBoolean, IsDate, IsNumeric, IsSimpleValue; Validating data with the IsValid function and the cfparam tag in the Developing ColdFusion Applications

History

ColdFusion 11: Behavioral change

Prior to ColdFusion 11, this function allowed currency symbols at the start and commas inside the number. Starting from ColdFusion 11, this function evaluates on a more strict basis. To revert to the old behavior use the application-level setting STRICTNUMBERVALIDATION to false.

ColdFusion 8: Added the component value for to the type attribute.

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>The valid format for the data; one of the following. For detailed information on validation algorithms, see Validating form data using hidden fields in the Developing ColdFusion Applications.</td>
</tr>
</tbody>
</table>
any: any simple value. Returns false for complex values, such as query objects; equivalent to the `IsSimpleValue` function.

array: an ColdFusion array; equivalent to the `IsArray` function.

binary: a binary value; equivalent to the `IsBinary` function.

boolean: a Boolean value: yes, no, true, false, or a number; equivalent to the `IsBoolean` function.

component: a ColdFusion component (CFC).

creditcard: a 13-16 digit number conforming to the mod10 algorithm.

date or time: any date-time value, including dates or times; equivalent to the `IsNumericDateIsDate` function.

e-mail: a valid email address.

eurodate: any date-time value, including US date formats and time values,

umeric: a numeric value; equivalent to the `IsNumeric` function.

guid: a Universally Unique Identifier of the form "XXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" where X is a hexadecimal number.

integer: an integer.

query: a query object; equivalent to the `IsQuery` function.

range: a numeric range, specified by the `min` and `max` attributes.

regex or regular_expression: matches input against pattern attribute.

ssn or social_security_number: A U.S. social security number.

string: a string value, including single characters and numbers

struct: a structure; equivalent to the `IsStruct` function.

television: a standard US telephone number.

URL: an http, https, ftp, file, mailto, or news URL,

UUID: a ColdFusion Universally Unique Identifier, formatted xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx, where x is a hexadecimal number. See Create UUID.

USdate: a U.S. date of the format mm/dd/yy, with 1-2 digit days and months, 1-4 digit years.

variableName: a string formatted according to ColdFusion variable naming conventions.

zipcode: U.S., 5- or 9-digit format ZIP codes.

| value | The value to test |
min

The minimum valid value; used only for range validation.

max

The maximum valid value; used only for range validation.

pattern

A JavaScript regular expression that the parameter must match; used only for regex or regular expression validation.

Usage

The `isValid` function lets you assure that validation is performed on the server. You can use the `cfparam` tag to perform equivalent validation.

Example

The following example checks whether a user has submitted a numeric ID and a valid e-mail address and phone number. If any of the submitted values does not meet the validation test, it displays an error message.

```
<cfif isDefined("form.saveSubmit")>
  <cfif isValid("integer", form.UserID) and isValid("email", form.emailAddr)
    and isValid("telephone", form.phoneNo)>
    <cfoutput>
      <!--- Application code to update the database goes here --->
      <h3>The email address and phone number for user #Form.UserID#
    have been added</h3>
    </cfoutput>
  </cfif>
</cfelse>
<H3>You must supply a valid User ID, phone number, and email address.</H2>
</cfif>
<cfelse>
</cfif>

<cfform action="#CGI.SCRIPT_NAME#">
  User ID: <cfinput type="Text" name="UserID"><br>
  Phone:  <cfinput type="Text" name="phoneNo"><br>
  email:  <cfinput type="Text" name="emailAddr"><br>
  <cfinput type="submit" name="saveSubmit" value="Save Data"><br>
</cfform>
```

In ColdFusion 11, the `isValid` function behaves in a different way. Setting `strictnumbervalidation` to false makes the `isValid` function behave in a way just like in the previous versions (ColdFusion 10 or earlier). Starting from ColdFusion 11, a more strict approach will be followed for validation. However, if you need the `isValid` function behavior to be of ColdFusion 10, set the key/tag attribute `strictnumbervalidation` to false in application cfc/cfm.

```
<Application.cfm

  <cfapplication name="hello" STRICTNUMBERVALIDATION="false">
```
Application.cfc

```coldfusion
<cfcomponent>
  <cfscript>
    { this.STRICTNUMBERVALIDATION = false; }
  </cfscript>
</cfcomponent>
```
IsWDDX

Description

Determines whether a value is a well-formed WDDX packet.

Returns

True, if the value is a well-formed WDDX packet; False, otherwise.

Category

Decision functions, XML functions

Syntax

IsWDDX(value)

See also

Using WDDX in the Developing ColdFusion Applications

History

ColdFusion MX: Changed behavior: if the value parameter is not a WDDX packet, ColdFusion returns False. (In earlier releases, ColdFusion threw an error.)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A WDDX packet</td>
</tr>
</tbody>
</table>

Usage

This function processes a WDDX packet with a validating XML parser, which uses the WDDX Document Type Definition (DTD). To prevent CFWDDX deserialization errors, you can use this function to validate WDDX packets from unknown sources.

Example
<cfset packet="" 
<wddxPacket version='1.0'> 
<header></header> 
<data> 
<struct> 
<var name='ARRAY'> 
<array length='3'> 
<string>one</string> 
<string>two</string> 
</array> 
</var> 
<var name=NUMBER> 
<string>5</string> 
</var> 
<var name=STRING> 
<string>hello</string> 
</var> 
</struct> 
</data> 
</wddxPacket>"
>
<hr>
<xmp> 
<cfoutput>#packet#</cfoutput> 
</xmp> 
IsWDDX() returns #IsWDDX(packet)#<br>
</cfoutput>
IsXML

Description

Determines whether a string is well-formed XML text.

Returns

True, if the function parameter is a string that contains well-formed XML text; False, otherwise.

Category

Decision functions, XML functions

Function syntax

IsXML(value)

See also

IsXmlAttribute, IsXmlDoc, IsXmlElem, IsXmlNode, IsXmlRoot, XmlParse, XmlValidate; Using XML and WDDX in the Developing ColdFusion Applications

History

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A string containing the XML document text</td>
</tr>
</tbody>
</table>

Usage

This function determines whether text is well-formed XML, that is, it conforms to all XML syntax and structuring rules. The string does not have to be a complete XML document. The function does not validate against a Document Type Definition (DTD) or XML Schema.

Example

The following example creates two strings, and tests whether they are well-formed XML text:
<!--- A well formed XML string --->
<cfset xmlString1='"<order id="4323251">
  <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
  <items>
    <item id="43">
      <quantity>1</quantity>
      <unitprice>15.95</unitprice>
    </item>
  </items>
</order>">

<!--- An invalid XML string, missing the </item> close tag --->
<cfset xmlString2='"<order id="4323251">
  <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
  <items>
    <item id="43">
      <quantity>1</quantity>
      <unitprice>15.95</unitprice>
    </items>
</order>''>

<!--- Test the strings to see if they are well formed XML --->
<cfoutput>
xmString1 contains the following text:<br><br>#HTMLCodeFormat(xmlstring1)#
Is it well formed XML text? #IsXML(xmlString1)#<br><br><hr>
xmString2 contains the following text:<br><br>#HTMLCodeFormat(xmlstring2)#
Is it well formed XML text? #IsXML(xmlString2)#</cfoutput>
IsXmlAttribute

Description

Determines whether the function parameter is an XML Document Object Model (DOM) attribute node.

Returns

True, if the function argument is an XML attribute node; False, otherwise.

Category

Decision functions, XML functions

Function syntax

IsXmlAttribute(value)

See also

IsXML, IsXmlDoc, IsXmlElem, IsXmlNode, IsXmlRoot, XmlGetNodeType, XmlValidate, Using XML and WDDX in the Developing ColdFusion Applications

History

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Name of an XML attribute</td>
</tr>
</tbody>
</table>

Usage

This function determines whether the parameter is an XML DOM attribute node, a node with an XMLType value of ATTRIBUTE. It is useful for determining whether a value returned by the XmlSearch function is an XML attribute. The DOM, and therefore ColdFusion, treats XML attributes as properties of an element and does not directly expose them as DOM nodes. For this reason, the XmlAttributes entries in ColdFusion XML document objects do not represent DOM attribute nodes, and tests such as the following always return False:

```
IsXmlAttribute(myxmlelement.XMLAttributes);
IsXmlAttribute(myxmlelement.XMLAttributes.myattribute);
```

The XmlSearch function does return attributes as XML DOM attribute nodes. For example, the following line returns an array of attribute nodes containing the quantity attributes in the xmlobject document object:

```
quantities = XmlSearch(xmlobject, '//@quantity');
```
Example

The following example creates an XML document object and gets parts of it. It then tests whether these parts are attribute nodes.

```xml
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
  <order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
      <item id="43">
        <quantity>1</quantity>
        <unitprice>15.95</unitprice>
      </item>
    </items>
  </order>
</cfxml>

<!--- Get an array with all lastname quantity DOM attribute nodes
(In this example there is only one entry) --->
<cfset lastnames = XmlSearch(xmlobject, '//@lastname')>

<!--- Test objects to see if they are attributes --->
<cfoutput>
  <h3>Are the following XML Attribute nodes?</h3>
  <!--- The order element id attribute.
  This a simple variable, not a DOM attribute node.--->
  node.xmlobject.order.XmlAttributes.id: #IsXmlAttribute(xmlobject.order.XmlAttributes.id)#<br>
  <!--- The items element --->
  xmlobject.order.items: #IsXmlAttribute(xmlobject.order.items)#<br>
  lastnames[1] returned by XmlSearch:
  #isXmlAttribute(lastnames[1])#<br>
</cfoutput>
```
IsXmlDoc

Description
Determine whether the function parameter is a ColdFusion XML document object.

Returns
True, if the function argument is an XML document object; False, otherwise.

Category
Decision functions, XML functions

Function syntax

IsXmlDoc(value)

See also
IsXML, IsXmlAttribute, IsXmlElem, IsXmlNode, IsXmlRoot, XmlValidate; Using XML and WDDX in the Developing ColdFusion Applications

History
ColdFusion MX: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Name of an XML document object</td>
</tr>
</tbody>
</table>

Example

The following example creates an XML Document object and a Java object and tests whether they are XML document objects:
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
<order id="4323251">
  <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
  <items>
    <item id="43">
      <quantity>1</quantity>
      <unitprice>15.95</unitprice>
    </item>
  </items>
</order>
</cfxml>

<!--- Create a Java object --->
<cfobject type="JAVA" action="create" class="java.lang.Error" name="javaobject">

<!--- Test the objects --->
<cfoutput>
Is xmlobject an XML document object? #IsXmlDoc(xmlobject)#<br>
Is javaobject an XML document object? #IsXmlDoc(javaobject)#<br>
</cfoutput>
IsXmlElem

Description

Determines whether the function parameter is an XML document object element.

Returns

True, if the function argument is an XML document object element; False, otherwise.

Category

Decision functions, XML functions

Function syntax

IsXmlElem(value)

See also

IsXML, IsXmlAttribute, IsXmlDoc, IsXmlNode, IsXmlRoot, XmlGetNodeType, XmlValidate; Using XML and WDDX in the Developing ColdFusion Applications

History

ColdFusion MX: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Name of an XML document object element</td>
</tr>
</tbody>
</table>

Example

The following example tests whether an XML document object, the document root, and an element are elements:
<!--- Create an XML document object --->
<cfxml variable="xmllobject">
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
    </items>
</order>
</cfxml>

<!--- Test parts of the document object to see if they are elements --->
<cfoutput>
    <h3>Are the following XML document object elements?</h3>
    xmllobject: #IsXmlElem(xmllobject)#<br>
    xmllobject.XMLRoot: #IsXmlElem(xmllobject.XMLRoot)#<br>
    xmllobject.order.items: #IsXmlElem(xmllobject.order.items)#<br>
</cfoutput>
IsXmlNode

Description

Determines whether the function parameter is an XML document object node.

Returns

True, if the function argument is an XML document object node, including an element; False, otherwise.

Category

Decision functions, XML functions

Function syntax

IsXmlNode(value)

See also

IsXML, IsXmlAttribute, IsXmlDoc, IsXmlElem, IsXmlRoot, XmlGetNodeType, XmlSearch, XmlValidate; Using XML and WDDX in the Developing ColdFusion Applications

History

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Name of an XML document object node.</td>
</tr>
</tbody>
</table>

Usage

This function returns True for the following components of an XML document object:

- The document object
- Elements in the object
- XmlNode objects in an element's XmlNodes arrayIt also returns True for XML node objects returned by the XmlSearch function. It does not return True for most entries in an element, including XmlText, XmlComment, XmlCdata, or the XmlAttributes array (or individual XML attributes).

Example

The following example tests whether an XML document object, an element, an attribute in the object, and an attribute returned by an XmlSearch function are nodes:
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
    <?xml version="1.0" encoding="UTF-8"?>
    <order id="4323251">
        <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
        <items>
            <item id="43">
                <quantity>1</quantity>
                <unitprice>15.95</unitprice>
            </item>
        </items>
    </order>
</cfxml>

<!--- use XmlSearch to get an attribute node. --->
<cfset lastnames = XmlSearch(xmlobject, '//@lastname')>

<!--- Test the objects to see if they are XML nodes--->
<coutput>
    <h3>Are the following XML nodes?</h3>
    xmlobject: #IsXmlNode(xmlobject)#
    xmlobject.order.items: #IsXmlNode(xmlobject.order.items)#
    xmlobject.order.XmlAttributes.id: #IsXmlNode(xmlobject.order.XmlAttributes.id)#
    lastnames[1] returned by XmlSearch:
    #isXmlNode(lastnames[1])#
</coutput>
IsXmlRoot

Description

Determines whether the function parameter is the root element of an XML document object.

Returns

True, if the function argument is the root object of an XML document object; False, otherwise.

Category

Decision functions, XML functions

Function syntax

IsXmlRoot(value)

See also

IsXML, IsXmlAttribute, IsXmlDoc, IsXmlElem, IsXmlNode, XmlGetNodeType, XmlValidate; Using XML and WDDX in the Developing ColdFusion Applications

History

ColdFusion MX: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Name of an XML document object</td>
</tr>
</tbody>
</table>

Example

The following example tests whether an XML document object, its root element, and a child element are XML root elements:
<!---- Create an XML document object --->
<cfxml variable="xmlobject">
<?xml version="1.0" encoding="UTF-8"?>
<order id="4323251">
    <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
    <items>
        <item id="43">
            <quantity>1</quantity>
            <unitprice>15.95</unitprice>
        </item>
    </items>
</order>
</cfxml>

<!---- Test objects to see if they are XML root elements --->
<cfoutput>
<h3>Are the following the XML Root?</h3>
xmlobject: #IsXmlRoot(xmlobject)#<br>
xmlobject.order: #IsXmlRoot(xmlobject.order)#<br>
<cfoutput>
<h3>The order element id attribute ---></h3>
xmlobject.order.XmlAttributes.id:
    #IsXmlRoot(xmlobject.order.XmlAttributes.id)#<br>
</cfoutput>
JavaCast

Description

Converts the data type of a ColdFusion variable to a specified Java type to pass as an argument to Java or .NET object. Use only for scalar, string, and array arguments.

Returns

The variable, as type type.

Category

String functions

Function syntax

JavaCast(type, variable)

History

ColdFusion MX 8: Added support for BigDecimal, byte, char, and short data types and for casting Arrays.
ColdFusion MX 7: Added support for nulls.

See also

CreateObject, cfobject, Converting between .NET and ColdFusion data types in Using .NET classes and Java and ColdFusion data type conversions in Using Java objects in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Data type to which to convert variable:</td>
</tr>
<tr>
<td></td>
<td>• BigDecimal (converts to java.math.BigDecimal)</td>
</tr>
<tr>
<td></td>
<td>• boolean</td>
</tr>
<tr>
<td></td>
<td>• byte</td>
</tr>
<tr>
<td></td>
<td>• char</td>
</tr>
<tr>
<td></td>
<td>• int</td>
</tr>
<tr>
<td></td>
<td>• long</td>
</tr>
<tr>
<td></td>
<td>• float</td>
</tr>
<tr>
<td></td>
<td>• double</td>
</tr>
<tr>
<td></td>
<td>• short</td>
</tr>
<tr>
<td></td>
<td>• string</td>
</tr>
<tr>
<td></td>
<td>• null</td>
</tr>
<tr>
<td></td>
<td>• xxx where xxx is one of the following:</td>
</tr>
<tr>
<td></td>
<td>• any of the preceding types, except for null</td>
</tr>
<tr>
<td></td>
<td>• a Java class name</td>
</tr>
<tr>
<td>variable</td>
<td>A ColdFusion variable that holds a scalar or string type. Must be &quot;&quot; if type is null.</td>
</tr>
</tbody>
</table>
Usage

Use this method to specify the Java type to use for a variable that you use when calling a Java or .NET method when the conversion between types is ambiguous; for example, if a method is overloaded and differs only in parameter type or a .NET method is declared as taking a System.Object class parameter. Use after creating a Java object with the `cfobject` tag, before calling one of its methods. If the method takes more than one overloaded argument, call `JavaCast` for each one. Use `JavaCast` only when a method is overloaded (because its arguments can take more than one data type, not because the method can take a variable number of arguments). `JavaCast` cannot be used to cast between complex objects, nor to cast to a super-class. Because there is not a one-to-one correspondence between internally stored ColdFusion types and Java scalar types, some conversions cannot be performed. Use the result of this function only on calls to Java or .NET objects. The following example shows the use when calling a Java method.

```cfscript
  <cfscript>
    x = CreateObject("java", "test.Hello");
    x.init();
    ret = x.sayHello(JavaCast("null", ""));
  </cfscript>
```

**Note**

Do not assign the results of `JavaCast("null", "")` to a ColdFusion variable. Unexpected results will occur.

The format `JavaCast(type[], variable)` casts a ColdFusion Array variable to a single dimensional Array of the specified type. It cannot convert multi-dimensional arrays. You can specify a primitive type or the name of a Class as the type to cast to. For example, you can use the following format to cast a ColdFusion Array to an Array of `vom.x.y.MyClass` objects.

```
javacast("vom.x.y.MyClass[]", myCFArr)
```

Use an array in the first `JavaCast` parameter in any of the following circumstances:

- You have two functions with signatures with the same number of parameters, and a parameter takes different types of Arrays in different signatures; for example, if you have both of the following functions: `foo(int[] x)` and `foo(String[] strs)`.
- The method parameter requires a class array in its signature; for example, `foo(com.x.y.MyClass[])`.
- The method parameter requires an Object in its signature and you must pass an array of any particular type. The following example shows the use of the `JavaCast` function to cast arrays: You might have a `fooClass` class that defines the following two methods, each with two arguments where the first argument differs in the type of the array:
public class fooClass {
    public fooClass () {
    }
    public String foo(long[] arg) {
        return "Argument was a long array";
    }
    public String foo(int[] arg) {
        return "Argument was an Integer array";
    }
}

To be able to use these functions in your CFML, use the `JavaCast` function to convert the ColdFusion Array to the array type required by one of the functions, as shown in the following code snippet:

```cfml
<cfset arr = [1,2,4,20,10]>
<cfset fooObj = createObject("java", "fooClass")>
<cfset fooObj.foo(javaCast("int[]", arr))>
<cfset fooObj.foo(javaCast("long[]", arr))>
```

Example

The method `fooMethod` in the class `fooClass` takes one overloaded argument. The `fooClass` class is defined as follows:

```java
public class fooClass {
    public fooClass () {
    }
    public String fooMethod(String arg) {
        return "Argument was a String";
    }
    public String fooMethod(int arg) {
        return "Argument was an Integer";
    }
}
```

Within ColdFusion, you use the following code:
<cfobject
action="create"
type = "java"
class = "fooClass"
name = obj>

<!--- ColdFusion can treat this as a string or a real number --->
<cfset x = 33>

Perform an explicit cast to an int and call fooMethod:<br>
<cfset myInt = JavaCast("int", x)>  
<cfoutput>#obj.fooMethod(myInt)#</cfoutput>

<br><br>
Perform an explicit cast to a string and call fooMethod:<br>
<cfset myString = javaCast("String", x)>  
<cfoutput>#obj.fooMethod(myString)#</cfoutput>
JSStringFormat

Description

Escapes special JavaScript characters, such as single-quotation mark, double-quotation mark, and newline.

Returns

A string that is safe to use with JavaScript.

Category

String functions

Function syntax

```
JSStringFormat(string)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one.</td>
</tr>
</tbody>
</table>

Usage

Escapes special JavaScript characters, so you can put arbitrary strings safely into JavaScript.

Example

```html
<!---- This example shows the use of the JSStringFormat function. ---->
<h3>JSStringFormat</h3>
<cfset stringValue = "An example string value with a tab chr(8),
   a newline (chr10) and some ""quoted"" 'text'">

<p>This is the string we have created:<br>
<cfoutput>#stringValue#</cfoutput></p>
<cfset jsStringValue = JSStringFormat(#stringValue#)>
<!----- Generate an alert from the JavaScript string jsStringValue. ---->

<SCRIPT>
  s = "<cfoutput>#jsStringValue#</cfoutput>";
  alert(s);
</SCRIPT>
```
**isSafeHTML**

This function validates for allowed HTML according to the rules specified in the antisamy policy file. This can be used to prevent unwanted user supplied HTML being used in an application.

**Returns**

Returns false if the input is violating the allowed HTML rules.

**Parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputString</td>
<td>Required. The string to be encoded.</td>
</tr>
<tr>
<td>PolicyFile</td>
<td>Optional. File path for antisamy policy file which specifies HTML rules. In case if not specified, there is a provision to set this at application level. Else the default policy file shipped with ColdFusion will be used.</td>
</tr>
<tr>
<td>throwOnError</td>
<td>Optional. If set to true, exception is thrown.</td>
</tr>
</tbody>
</table>

**Example**

This example illustrates a sample Antisamy component.

```cfcomponent>
  <cfset this.security.antisamypolicy = "antisamy.xml">
</cfcomponent>
```

The following example illustrates a isSafeHTML and getSafeHTML validation.

In the following example, the input HTML file is a GIF image ([http://www.techsite.com/img.gif](http://www.techsite.com/img.gif)). On processing it through the functions (isSafeHTML, getSafeHTML) checks if the given input string violates the rules specified in the Antisamy policy file and returns true if the input does not violate the allowed HTML rules.

```cfset inputHTML= "<img src='http://www.techsite.com/img.gif'/>">
<cfset isSafe = isSafeHTML(inputHTML)>
<cfset SafeHTML = getSafeHTML(inputHTML, "", true)>
<cfoutput>
is Safe  : #
isSafe#  
Safe HTML  : #
Safe HTML#</cfoutput>```
InvalidateOauthAccessToken

Description
Invalidates the access token generated by the OAUTH servers.

Returns
None

Function syntax

```
InvalidateOauthAccessToken(token, type)
```

History
ColdFusion 11: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>Required. The access token to invalidate.</td>
</tr>
<tr>
<td>type</td>
<td>Required. The type of the OAUTH server (facebook/google).</td>
</tr>
</tbody>
</table>

Usage

Example

```
InvalidateOauthAccessToken("any_token_value","facebook")
```
IsValidOauthAccessToken

Description
Checks if the OAUTH access token is still valid.

Returns
Boolean

Function syntax
IsValidOauthAccessToken(token, type)

History
ColdFusion 11: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>Required. The access token to check.</td>
</tr>
<tr>
<td>type</td>
<td>Required. The type of the OAUTH server (facebook/google).</td>
</tr>
</tbody>
</table>

Usage

Example
IsValidOauthAccessToken("any_token_value","facebook")
InvokeCFClientFunction

Description

ColdFusion automatically determines whether a function call is synchronous or asynchronous. However, if you need to invoke an asynchronous function in a synchronous mode, you can use the invokeInSyncMode function. The function call just needs to be wrapped around with the invokeInSyncMode function call. For instance, invokeInSyncMode (myAsyncFunc(arg1,arg2)).

The invokeCFClientFunction function can be used to invoke both synchronous and asynchronous CFML functions from JavaScript.

History

ColdFusion 11: Added this function

Syntax

```javascript
invokeCFClientFunction(funcName, arg1, arg2, ..., successCallback)
```

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>funcname</td>
<td>Required. The function to be invoked.</td>
</tr>
<tr>
<td>arguments</td>
<td>Optional. Arguments to pass to the function.</td>
</tr>
<tr>
<td>succesCallback</td>
<td>The callback handler when the function gets invoked successfully. The succesCallback argument is mandatory. If there is no succesCallback, use null.</td>
</tr>
</tbody>
</table>

Example

The following code depicts a simple usage of file creation on the device. Since file creation is an asynchronous operation, we are using the invokeCFClientFunction to invoke this function.
<cfclientsettings enableDeviceAPI=true>
<cfclient>

<cffunction access="public" name="createfile" returntype="void"/>
</cffunction>

</cfclient>

File name : <input id="filename" type="text"/>

<button onclick="invokeCFClientFunction('createfile',null)">Create a file</button>
<br>
<b>Result :</b><div id="result"></div>

<script type="text/javascript">
function showresult(obj)
{
    return JSON.stringify(obj);
}
</script>

For more information on using this function, see this community blog post.
Functions I

- LCase
- Left
- Len
- ListAppend
- ListChangeDelims
- ListContains
- ListContainsNoCase
- ListDeleteAt
- ListFilter
- ListFind
- ListFindNoCase
- ListFirst
- ListGetAt
- ListInsertAt
- ListLast
- ListLen
- ListPrepend
- ListQualify
- ListRemoveDuplicates
- ListRest
- ListSetAt
- ListSort
- ListToArray
- ListEach
- ListReduce
- ListMap
**LCase**

**Description**

Converts the alphabetic characters in a string to lowercase.

**Returns**

A string, converted to lowercase.

**Category**

String functions

**Function syntax**

```
LCase(string)
```

**See also**

UCase

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

**Example**

```html
<h3>LCase Example</h3>
<cfif IsDefined("FORM.sampleText")>
  <cfif FORM.sampleText is not ">"
    <coutput>
      <p>Your text, <b>#FORM.sampleText#</b>, returned in lowercase is <b>#LCase(FORM.sampleText)#</b>.</p>
    </coutput>
  </cfif>
  <cfelse>
    <p><b><i>Please enter some text.</i></b></p>
  </cfelse>
</cfif>

<p>Enter your text. Press "submit" to see it returned in lowercase: </p>
<form method="post" action="#cgi.script_name#" name="lcaseForm">
  <input type = "Text" name = "SampleText" value = "SAMPLE">
  <input type = "Submit" name = "" value = "submit">
</form>
```
Left

Description

Returns up to the leftmost `count` characters in a string.

Returns

String; up to the first `count` characters in the `string` parameter.

Category

String functions

Function syntax

```
Left(string, count)
```

See also

`Right`, `Mid`, `Len`

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string</code></td>
<td>A string or a variable that contains one.</td>
</tr>
<tr>
<td><code>count</code></td>
<td>A positive integer or a variable that contains one. Maximum number of characters to return.</td>
</tr>
</tbody>
</table>

Example
<h3>Left Example</h3>

<cfif IsDefined("Form.myText")>
<!---- If len returns 0 (zero), then show error message. --->
<cfif Len(Form.myText)>
<cfif Len(Form.myText) LTE Form.RemoveChars>
<cfoutput><p style="color: red; font-weight: bold">Your string #Form.myText# only has #Len(Form.myText)# characters. You cannot output the #Form.removeChars# leftmost characters of this string because it is not long enough.</p></cfoutput>
<cfelse>
<cfoutput><p>Your original string: <strong>#Form.myText#</strong></p>
<p>Your changed string, showing only the <strong>#Form.removeChars#</strong> leftmost characters:<br>  <strong>#Left(Form.myText, Form.removeChars)#</strong></p>
</cfoutput>
</cfif>
</cfif>
<cfelse>
<p style="color: red; font-weight: bold">Please enter a string of more than 0 (zero) characters.</p>
</cfif>
</cfif>

<form action="<cfoutput>#CGI.ScriptName#</cfoutput>" method="POST">
<p>Type in some text<br /></p>
<input type="Text" name="myText"/>
<p>How many characters from the left do you want to show?<br></p>
<select name="RemoveChars">
<option value="1">1</option>
<option value="3" selected>3</option>
<option value="5">5</option>
<option value="7">7</option>
<option value="9">9</option>
</select>
<input type="Submit" name="Submit" value="Remove characters"/>
</form>
Len

Description

Determines the length of a string or binary object.

Returns

Number; length of a string or a binary object.

Category

String functions

Function syntax

Len(string or binary object)

See also

ToBinary, Left, Right, Mid

History

ColdFusion MX: Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode character values 0-65535. (ColdFusion 5 and earlier releases supported ASCII values 1-255. When calculating a length, some string-processing functions processed the ASCII 0 (NUL) character, but did not process subsequent characters of the string.)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string, the name of a string, or a binary object</td>
</tr>
</tbody>
</table>

Example
<h3>Len Example</h3>

<cfif IsDefined("Form.MyText")>
<!--- If len returns 0 (zero), then show error message. --->
<cfif Len(FORM.myText)>
<coutput>
<p>Your string, <strong>#FORM.myText#</strong>, has <strong>#Len(FORM.myText)#</strong> characters.</p>
</coutput>
<cfelse>
<p style="color: red; font-weight: bold">Please enter a string of more than 0 characters.</p>
</cfelse>
</cfif>

<form action="#CGI.SCRIPT_NAME#" method="POST">
<p>Type in some text to see the length of your string.</p>
<input type = "Text" name = "MyText"/><br />
<input type = "Submit" name="Submit" value = "Count characters"/><br />
</form>
**ListAppend**

**Description**

Concatenates a list or element to a list.

**Returns**

A copy of the list, with `value` appended. If `value = ""`, returns a copy of the list, unchanged.

**Category**

*List functions*

**Function syntax**

```
ListAppend(list, value [, delimiters ])
```

**See also**

`ListPrepend`, `ListInsertAt`, `ListGetAt`, `ListLast`, `ListSetAt`; *Lists* in the *Developing ColdFusion Applications*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>value</td>
<td>An element or a list of elements.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion uses only the first character.</td>
</tr>
</tbody>
</table>

**Usage**

ColdFusion inserts a delimiter character before `value`. The following table shows examples of `ListAppend` processing:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Output</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListAppend('elem1,elem2', '' )</td>
<td>elem1,elem2,</td>
<td>Appended element is empty; delimiter is last character in list; list length is 2.</td>
</tr>
<tr>
<td>ListAppend('', 'elem1,elem2' )</td>
<td>elem1,elem2</td>
<td>List length is 2.</td>
</tr>
</tbody>
</table>
| ListAppend("one__two",  
| "three", "__") | "one__two_three" | Inserted the first character of delimiters before "three." |

Example

```coldfusion
<h3>ListAppend Example</h3>
<cfquery name = "GetParkInfo" datasource = "cfdocexamples">
  SELECT PARKNAME,CITY,STATE 
  FROM PARKS WHERE PARKNAME LIKE 'AL%'
</cfquery>
<cfset temp = ValueList(GetParkInfo.ParkName)>
<cfoutput>
<p>The original list: #temp#</p>
</cfoutput>
<cfset temp2 = ListAppend(Temp, "ANOTHER PARK")>
```
ListChangeDelims

Description

Changes a list delimiter.

Returns

A copy of the list, with each delimiter character replaced by new_delimiter.

Category

List functions

Function syntax

ListChangeDelims(list, new_delimiter [, delimiters, includeEmptyValues ])

See also

ListFirst, ListQualify; Lists in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>new_delimiter</td>
<td>Delimiter string or a variable that contains one. Can be an empty string.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

Example
<h3>ListChangeDelims Example</h3>

ListChangeDelims lets you change the delimiters of a list.

<!--- First, query to get some values for our list elements--->

```coldfusion
<CFQUERY NAME="GetParkInfo" DATASOURCE="cfdocexamples">
    SELECT PARKNAME, CITY, STATE
    FROM Parks
    WHERE PARKNAME LIKE 'BA%'
</CFQUERY>
```

```coldfusion
<CFSET temp = ValueList(GetParkInfo.ParkName)>
<cfoutput>
The original list: <p>#temp#
</cfoutput>

<!--- Change the delimiters in the list --->
<cfset temp2 = ListChangeDelims(Temp, "|:P|", ",")>
<cfoutput>
After executing the statement
<br>&lt;strong&gt;ListChangeDelims(Temp, "|:P|", ",")&lt;/strong&gt;,  
the updated list: <p>#temp2#
</cfoutput>
**ListContains**

**Description**

Determines the index of the first list element that contains a specified substring.

**Returns**

Index of the first list element that contains *substring*. If not found, returns zero.

**Category**

List functions

**Function syntax**

```
ListContains(list, substring [, delimiters, includeEmptyValues ])
```

**See also**

[ListContainsNoCase](#), [ListFind](#); Lists in [Data types- Developing guide](#) in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to <em>yes</em> to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>substring</td>
<td>A string or a variable that contains one. The search is case sensitive.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

**Usage**

ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

**Example**
<!--- This example shows differences between ListContains and ListFind --->
<!--- Create a list composed of the elements one, two, three. ---->
<cfset aList = "one">
<cfset aList = ListAppend(aList, "two")>
<cfset aList = ListAppend(aList, "three")>
<p>Here is a list: <cfoutput>#aList#</cfoutput></p>
<p><strong>ListContains</strong> checks for substring "wo" in the list elements:
<cfoutput>
&nbsp;&nbsp;&nbsp;Substring "wo" is in <B>element #ListContains(aList, "wo")#</B> of list.
</cfoutput></p>
<p>ListFind cannot check for a substring within an element; therefore, in the code, it does not find substring "wo" (it returns 0):
<cfoutput>
&nbsp;&nbsp;&nbsp;Substring "wo" is in <b>element #ListFind(aList, "wo")#</b> of the list.
</cfoutput></p>
<p>If you specify a string that exactly equals an entire list element, such as "two", both ListContains and ListFind find it, in the second element:
<cfoutput>
Substituting "two" in <b>element #ListContains(aList, "two")#</b> of the list.
</cfoutput>
</cfoutput></p>
<p>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
**ListContainsNoCase**

**Description**

Determines the index of the first list element that contains a specified substring.

**Returns**

Index of the first list element that contains `substring`, regardless of case. If not found, returns zero.

**Category**

List functions

**Function syntax**

```
ListContainsNoCase(list, substring [, delimiters, includeEmptyValues ])
```

**See also**

ListContains, ListFindNoCase; Lists in Data types- Developing guide in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>substring</td>
<td>A string or a variable that contains one. The search is case-insensitive.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

**Usage**

ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

**Example**
<h3>ListContainsNoCase Example</h3>
<cfif IsDefined("form.letter")>
<!---- First, query to get some values for our list --->
<cfquery name="GetParkInfo" datasource="cfdocexamples">
  SELECT PARKNAME,CITY,STATE
  FROM Parks
  WHERE PARKNAME LIKE '#form.letter#%'
</cfquery>
<cfset tempList = #ValueList(GetParkInfo.City)#>
<cfif ListContainsNoCase(tempList, form.yourCity) is not 0>
There are parks in your city!
</cfif>
<cfelse>
<p>Sorry, there were no parks found for your city.
Try searching under a different letter.</p>
</cfif>
</cfif>
**ListDeleteAt**

**Description**

Deletes an element from a list.

**Returns**

A copy of the list, without the specified element.

**Category**

List functions

**Function syntax**

```
ListDeleteAt(list, position [, delimiters ])
ListDeleteAt(list, position [, delimiters, includeEmptyValues ])
```

**See also**

ListGetAt, ListSetAt, ListLen; Lists in Data types- Developing guide in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>position</td>
<td>A positive integer or a variable that contains one.</td>
</tr>
<tr>
<td></td>
<td>Position at which to delete element. The first list position is 1.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
<tr>
<td>includeEmptyValues</td>
<td>If includeEmptyValues is set to false, the list will contain only nonempty elements and the element would be deleted at the specified position.</td>
</tr>
</tbody>
</table>

**Usage**

To use this and other functions with the default delimiter (comma), you can code as follows:

```
<cfset temp2 = ListDeleteAt(temp, "3")>
```
To specify another delimiter, you code as follows:

```cfml
<cfset temp2 = ListDeleteAt(temp, "3", ";")>
```

ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

Example

```cfml
<!--- First, query to get some values for our list elements. --->
<CFQUERY NAME="GetParkInfo" DATASOURCE="cfdocexamples">
  SELECT PARKNAME,CITY,STATE
  FROM Parks
  WHERE PARKNAME LIKE 'CHI%'
</CFQUERY>
<CFSET temp = ValueList(GetParkInfo.ParkName)>
<CFSET deleted_element = ListGetAt(temp, "3", ",")>
<cfoutput><p>The original list: #temp#</p></cfoutput>
<!--- Delete the third element from the list. --->
<CFSET temp2 = ListDeleteAt(Temp, "3")>
<cfoutput>
  <p>The changed list: #temp2#</p>
  <p>This list element: <br><b>#deleted_element#</b> is no longer present at position three of the list.</p>
</cfoutput>
writeOutput(listDeleteAt("1,,3", 2, ",", true));//outputs 1,3
```
**ListFilter**

**Description**

Used to filter the elements in list.

**Returns**

A new list

**Category**

Closure functions

**Syntax**

```javascript
listFilter(list, function(listElement){return true|false;});
```

**See also**

Other closure functions.

**History**

ColdFusion 10: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Name of the list object.</td>
</tr>
<tr>
<td>function</td>
<td>Inline function executed for each element in the list. Returns \texttt{true} if the list element has to be included in the resultant list.</td>
</tr>
<tr>
<td>listElement</td>
<td>List element being accessed.</td>
</tr>
</tbody>
</table>
ListFind

Description

Determines the index of the first list element in which a specified value occurs. Case sensitive.

Returns

Index of the first list element that contains value, with matching case. If not found, returns zero. The search is case sensitive.

Category

List functions

Function syntax

```
ListFind(list, value [, delimiters, includeEmptyValues ])
```

See also

ListContains, ListFindNoCase; Lists in Data types- Developing guide in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one</td>
</tr>
<tr>
<td>value</td>
<td>A string, a number, or a variable that contains one. Item for which to search. The search is case sensitive.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

Usage

ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

Example
<form action="/listfind.cfm" method="POST">
<p>Try changing the case in Leary's last name:</p>
<br><input type="text" size="25" name="myString" value="Leary">
<p>Pick a search type:</p>
<select name="type">
<option value="ListFind" selected>Case-Sensitive</option>
<option value="ListFindNoCase">Case-Insensitive</option>
</select>
<input type="submit" value="Search Employee List">
</form>

<!--- wait to have a string for searching defined --->
<cfif structKeyExists(form,"myString") AND structKeyExists(form,"type")>
<cfquery name="SearchEmpLastName" datasource="cfdocexamples">
SELECT FirstName, RTrim(LastName) AS LName, Phone, Department FROM Employees
</cfquery>
<cfset myList = ValueList(SearchEmpLastName.LName)>
<!--- Is this case-sensitive or case-insensitive searching --->
<cfif form.type EQ "ListFind">
<cfset temp = ListFind(myList, form.myString)>
<cfif temp EQ 0>
<h3>An employee with that exact last name was not found</h3>
<cfelse>
<cfoutput>
Employee #ListGetAt(ValueList(SearchEmpLastName.FirstName), temp)# #ListGetAt(ValueList(SearchEmpLastName.LName), temp)#, of the #ListGetAt(ValueList(SearchEmpLastName.Department), temp)# Department, can be reached at #ListGetAt(ValueList(SearchEmpLastName.Phone), temp)#.</cfoutput>
<p>This was the first employee found under this case-sensitive last name search.</p>
</cfoutput>
</cfif>
<cfelse>
<cfoutput>
Employee #ListGetAt(ValueList(SearchEmpLastName.FirstName), temp)# #ListGetAt(ValueList(SearchEmpLastName.LName), temp)#, of the #ListGetAt(ValueList(SearchEmpLastName.Department), temp)# Department, can be reached at #ListGetAt(ValueList(SearchEmpLastName.Phone), temp)#.</cfoutput>
<p>This was the first employee found under this case-insensitive last name search.</p>
</cfoutput>
</cfif>
<cfelse>
<h3>An employee with that exact last name was not found</h3>
</cfelse>
</cfif>
ListFindNoCase

Description

Determines the index of the first list element in which a specified value occurs.

Returns

Index of the first list element that contains value. If not found, returns zero. The search is case-insensitive.

Category

List functions

Function syntax

```
ListFindNoCase(list, value [, delimiters, includeEmptyValues ])
```

See also

ListContains, ListFind; Lists in Data types- Developing guide in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>value</td>
<td>Number or string for which to search. The search is case-insensitive.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

Usage

ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

Example
<!--- Uses ListFind and ListFindNoCase to see if a substring exists
in a list --->
<form action="./listfind.cfm" method="POST">
  <p>Try changing the case in Leary's last name:
  </p><input type="Text" size="25" name="myString" value="Leary">
  <p>Pick a search type:
  </p><select name="type">
    <option value="ListFind" selected>Case-Sensitive
    <option value="ListFindNoCase">Case-Insensitive
  </select>
  <input type="Submit" name="" value="Search Employee List">
</form>

<!--- wait to have a string for searching defined --->
<cfif IsDefined("form.myString") and IsDefined("form.type")>
<cfquery name="SearchEmpLastName" datasource="cfdocexamples">
  SELECT FirstName, RTrim(LastName) AS LName, Phone, Department
  FROM Employees
</cfquery>

<cfset myList = ValueList(SearchEmpLastName.LName)>
<!--- Is this case-sensitive or case-insensitive searching --->
<cfif form.type is "ListFind">
  <cfset temp = ListFind(myList, form.myString)>
  <cfif temp is 0>
    <h3>An employee with that exact last name was not found</h3>
  </cfif>
  <cfelse>
    <cfoutput>
      Employee #ListGetAt(ValueList(SearchEmpLastName.FirstName), temp)#
      #ListGetAt(ValueList(SearchEmpLastName.LName), temp)#, of the
      #ListGetAt(ValueList(SearchEmpLastName.Department), temp)# Department,
      can be reached at #ListGetAt(ValueList(SearchEmpLastName.Phone), temp)#.
    </cfoutput>
    <p>This was the first employee found under this case-sensitive last name search.</p>
  </cfelse>
</cfif>
<cfelse>
  <cfset temp = ListFindNoCase(myList, form.myString)>
  <cfif temp is 0>
    <h3>An employee with that exact last name was not found</h3>
  </cfif>
  <cfelse>
    <cfoutput>
      Employee #ListGetAt(ValueList(SearchEmpLastName.FirstName), temp)#
      #ListGetAt(ValueList(SearchEmpLastName.LName), temp)#, of the
      #ListGetAt(ValueList(SearchEmpLastName.Department), temp)# Department,
      can be reached at #ListGetAt(ValueList(SearchEmpLastName.Phone), temp)#.
    </cfoutput>
    <p>This was the first employee found under this case-insensitive last name search.</p>
  </cfelse>
</cfif>
</cfif>
**ListFirst**

**Description**

Gets the first element of a list.

**Returns**

The first element of a list. If the list is empty, returns an empty string.

**Category**

List functions

**Function syntax**

```
ListFirst(list [, delimiters, includeEmptyValues ])
```

**See also**

[ListGetAt, ListLast, ListQualify]: Lists in [Data types- Developing guide] in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains a list.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

**Usage**

ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

**Example**
<h3>ListFirst Example</h3>

<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
    SELECT Username, Subject, Posted
    FROM Messages
</cfquery>

<!--- Show the first user in the list --->
<p>The first user in the list is: <cfoutput>#ListFirst(temp)#</cfoutput>.</p>
<p>The rest of the list is: <cfoutput>#ListRest(temp)#</cfoutput>.</p>
<p>The last user in the list is: <cfoutput>#ListLast(temp)#</cfoutput>.</p>
ListGetAt

Description

Gets a list element at a specified position.

Returns

Value of the list element at position position.

Category

List functions

Function syntax

ListGetAt(list, position [, delimiters, includeEmptyValues ])

See also

ListFirst, ListLast, ListQualify, ListSetAt; Lists in Data types- Developing guide in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>position</td>
<td>A positive integer or a variable that contains one. Position at which to get element. The first list position is 1.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
</tbody>
</table>

Usage

If you use list functions on strings that are delimited by a delimiter character and a space, a returned list element might contain a leading space; you use the trim function to remove such spaces from a returned element. For example, consider this list:

```
<cfset myList = "one hundred, two hundred, three hundred">
```
To get a value from this list, use the `trim` function to remove the space before the returned value:

```coldfusion
<cfset MyValue = #trim(listGetAt(myList, 2))#>
```

With this usage, `MyValue` = "two hundred", not "two hundred", and spaces within a list element are preserved. ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

**Example**

```coldfusion
<h3>ListGetAt Example</h3>
<!---- Find a list of users who wrote messages ---->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
   SELECT Username, Subject, Posted
   FROM Messages
</cfquery>
<cfset temp = ValueList(GetMessageUser.Username)>
<!---- loop through the list and show it with ListGetAt ---->
<h3>This list of usernames who have posted messages numbers</h3>
<cfoutput>#ListLen(temp)#</cfoutput> users.</h3>
<ul>
<cfloop From = "1" To = "#ListLen(temp)#" index = "Counter">
   <cfoutput><li>Username #Counter#: #ListGetAt(temp, Counter)#</cfoutput>
</cfloop>
</ul>
```
**ListInsertAt**

**Description**

Inserts an element in a list.

**Returns**

A copy of the list with value inserted at the specified position.

**Category**

List functions

**Function syntax**

```coldfusion
ListInsertAt(list, position, value [, delimiters, includeEmptyValues ])
```

**See also**

ListDeleteAt, ListAppend, ListPrepend, ListSetAt; Lists in Data types- Developing guide in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>position</td>
<td>A positive integer or a variable that contains one. Position at which to insert element. The first list position is 1.</td>
</tr>
<tr>
<td>value</td>
<td>An element or a list of elements.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

**Usage**

When inserting an element, ColdFusion inserts a delimiter. If delimiters contains more than one delimiter, ColdFusion uses the first delimiter in the string; if delimiters is omitted, ColdFusion uses a comma. ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

**Example**
<!--- This example shows ListInsertAt --->
<cfquery name = "GetParkInfo" datasource = "cfdocexamples">
SELECT PARKNAME,CITY,STATE
FROM PARKS
WHERE PARKNAME LIKE 'DE%'
</cfquery>
<cfset temp = ValueList(GetParkInfo.ParkName)>
<cfset insert_at_this_element = ListGetAt(temp, "3", ",")>
<cfoutput>
<p>The original list: #temp#</p>
</cfoutput>
<cfset temp2 = ListInsertAt(Temp, "3", "my Inserted Value")>
**ListLast**

**Description**

Gets the last element of a list.

**Returns**

The last element of the list.

**Category**

List functions

**Function syntax**

```
ListLast(list [, delimiters, includeEmptyValues ])
```

**See also**

ListGetAt, ListFirst; Lists in Data types- Developing guide in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains a list.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter; you cannot specify a multicharacter delimiter.</td>
</tr>
</tbody>
</table>

**Usage**

If you use list functions on strings that separated by a delimiter character and a space, a returned list element might contain a leading space; use the trim function to remove leading and trailing spaces from a returned element. For example, consider this list:

```
<cfset myList = "one hundred, two hundred, three hundred">  
```

To get a value from this list, use the trim function to remove the space before the returned value:
With this usage, the MyValue variable gets the value "three hundred", not "three hundred", and spaces within a list element are preserved. ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

Example

```
<cfset MyValue = #trim(ListLast(myList)#>

<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
  SELECT Username, Subject, Posted
  FROM Messages
</cfquery>
<cfset temp = ValueList(GetMessageUser.Username)>
<p>Before editing the list, it is: #ValueList(GetMessageUser.Username)#.
<p>(Users who posted more than once are listed more than once.)
<!--- Show the first user in the list --->
<p>The first user in the list is: #ListFirst(temp)#
<p>The rest of the list is: #ListRest(temp)#.
<p>(Users who posted more than once are listed more than once.)
<p>The last user in the list is: #ListLast(temp)#
```
**ListLen**

**Description**

Determines the number of elements in a list. Integer; the number of elements in a list.

**Category**

List functions

**Function syntax**

```
ListLen(list [, delimiters ])
ListLen(list [, delimiters, includeEmptyValues ])
```

**See also**

ListAppend, ListDeleteAt, ListInsertAt, ListPrepend; Lists in Data types- Developing guide in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
<tr>
<td>includeEmptyValues</td>
<td>If includeEmptyValues is set to true, all empty values in the list will be considered when computing length. If set to false, the empty list elements are ignored.</td>
</tr>
</tbody>
</table>

**Usage**

ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements. Here are some examples of ListLen processing:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Output</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListLen('a,b, c,,d')</td>
<td>4</td>
<td>Third element is &quot;c&quot;</td>
</tr>
<tr>
<td>ListLen('a,b, c,,d''')</td>
<td>4</td>
<td>Fourth element is &quot;d'&quot;</td>
</tr>
<tr>
<td>ListLen('elem_1__elem_2__elem_3')</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
ListLen('elem*1**elem*2**elem*3') | 1
ListLen('elem_1__elem_2__elem_3', '') | 6
ListLen('1,,3,,', '', true) | 3

Example

```cfsoup
<h3>ListLen Example</h3>
<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
    SELECT Username, Subject, Posted
    FROM Messages
</cfquery>
<cfset temp = ValueList(GetMessageUser.Username)>
<!--- loop through the list and show it with ListGetAt --->
<h3>This is a list of usernames who have posted messages</h3>
<cfoutput>#ListLen(temp)#</cfoutput> users.</h3>
<ul>
    <cfloop From = "1" TO = "#ListLen(temp)#" INDEX = "Counter">
        <cfoutput><li>Username #Counter#:</li> #ListGetAt(temp, Counter)#</cfoutput>
    </cfloop>
</ul>
```
ListPrepend

Description

Inserts an element at the beginning of a list.

Returns

A copy of the list with value inserted at the first position.

Category

List functions

Function syntax

```
ListPrepend(list, value [, delimiters ])
```

See also

ListAppend, ListInsertAt, ListSetAt: Lists in Data types- Developing guide in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>value</td>
<td>An element or a list of elements.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion only uses the first character and ignores the others.</td>
</tr>
</tbody>
</table>

Usage

When prepending an element to a list, ColdFusion inserts a delimiter. If delimiters contains more than one delimiter character, ColdFusion uses the first delimiter in the string; if delimiters is omitted, ColdFusion uses a comma. ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements. If the delimiters parameter is the empty string (""), ColdFusion returns the contents of the value parameter.

Example
<!--- This example shows ListPrepend --->
<cfquery name = "GetParkInfo" datasource = "cfdocexamples">
    SELECT PARKNAME,CITY,STATE
    FROM PARKS
    WHERE PARKNAME LIKE 'DE%'
</cfquery>
<cfset temp = ValueList(GetParkInfo.ParkName)>
<cfset first_element = ListFirst(temp)>
<cfoutput><p>The original list: #temp#</cfoutput>
<!--- now, insert an element at position 1--->
<cfset temp2 = ListPrepend(Temp, "my Inserted Value")>
ListQualify

Description

Inserts a string at the beginning and end of list elements.

Returns

A copy of the list, with qualifier before and after the specified elements.

Category

List functions

Function syntax

```
ListQualify(list, qualifier [, delimiters, elements, includeEmptyValues ])
```

See also

Lists in Using ColdFusion Variables in the Developing ColdFusion Applications

History

ColdFusion MX: Changed behavior: as the elements parameter value, you must specify "all" or "char"; otherwise, ColdFusion throws an exception. (In earlier releases, the function ignored an invalid value, and used "all"; this was inconsistent with other functions.)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>qualifier</td>
<td>A string or a variable that contains one. Character or string to insert before and after the list elements specified in the elements parameter.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion uses the first character as the delimiter and ignores the remaining characters.</td>
</tr>
</tbody>
</table>
| elements                | • all: all elements  
                           • char: elements that are composed of alphabetic characters |

Usage
The new list might not preserve all of the delimiters in the list. ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

Example

```
<cfquery name = "GetEmployeeNames" datasource = "cfdocexamples">
    SELECT  FirstName, LastName
    FROM  Employees
</cfquery>

<h3>ListQualify Example</h3>
<p>This example uses ListQualify to put the full names of the employees in the query within quotation marks.</p>
<cfset myArray = ArrayNew(1)>
<!--- loop through query; append these names successively to the last element --->
<cfloop query = "GetEmployeeNames">
    <cfset temp = ArrayAppend(myArray, "#FirstName# #LastName#")>
</cfloop>
<!--- sort that array descending alphabetically ---->
<cfset myAlphaArray = ArraySort(myArray, "textnocase")>
<!--- show the resulting array as a list ---->
<cfset myList = ArrayToList(myArray, ",")>

<cfoutput>
    <p>The contents of the unqualified list are as follows: </p>
    #myList#
</cfoutput>

<!--- show the resulting alphabetized array as a qualified list with single quotation marks around each name. ---->
<cfset qualifiedList1 = ListQualify(myList,"'",",","CHAR")>

<!--- output the array as a list ---->
<cfoutput>
    <p>The contents of the qualified list are as follows: </p>
    #qualifiedList1#
</cfoutput>

<!--- show the resulting alphabetized array as a qualified list with quotation marks around each full name. We use &quot; to denote quotation marks because the quotation mark character is a control character. ---->
<cfset qualifiedList2 = ListQualify(myList,"&quot;","","CHAR")>

<!--- output the array as a list ---->
<cfoutput>
    <p>The contents of the second qualified list are: </p>
    #qualifiedList2#
</cfoutput>
```
ListRemoveDuplicates

Description

Removes duplicate values (if they exist) in a list.

Returns

List sans duplicate values

History

ColdFusion 10: Added this function

Syntax

ListRemoveDuplicates(list[, delimiter] [, ignoreCase])

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Required. List of objects.</td>
</tr>
<tr>
<td>delimiter</td>
<td>Optional. Character(s) that separate list elements. The default value is comma.</td>
</tr>
<tr>
<td>ignoreCase</td>
<td>Optional. If true, ignores the case of strings in the list. By default the value is set to false.</td>
</tr>
</tbody>
</table>

Example

```cfscript
<cfscript>
   myList = "one,two,three,four,five,one,five,three"
   newList = listremoveduplicates(myList);
   //default delimiter is ","
   //newList contains "one,two,three,four,five"
</cfscript>
```

```cfscript
<cfscript>
   myList = "one,two,three,four,five,ONE,TWO,THREE"
   newList = listremoveduplicates(myList, ",", true);
   //newList contains "one,two,three,four,five"
</cfscript>
```
ListRest

Description

Gets a list, without its first element.

Returns

A copy of list, without the first element. If list has one element, returns an empty list.

Category

List functions

Function syntax

ListRest(list [, delimiters, includeEmptyValues ])

See also

ListFirst, ListGetAt, ListLast: Lists in Data types- Developing guide in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

Usage

If the list begins with one or more empty entries, this function drops them, as well as the first element. ColdFusion ignores empty list elements; thus, the list "a,b,c,,d" has four elements.

Example
<h3>ListFirst, ListLast, and ListRest Example</h3>

<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
  SELECT Username, Subject, Posted
  FROM Messages
</cfquery>

<cfset temp = ValueList(GetMessageUser.Username)>
<p>Before editing the list, it is: &nbsp;#ValueList(GetMessageUser.Username)#</p>
<p>(Users who posted more than once are listed more than once.)</p>
<p>The first user in the list is: &nbsp;#ListFirst(temp)#</p>
<p>The rest of the list is: &nbsp;#ListRest(temp)#</p>
<p>(Users who posted more than once are listed more than once.)</p>
<p>The last user in the list is: &nbsp;#ListLast(temp)#</p>
ListSetAt

Description

Replaces the contents of a list element.

Returns

A copy of a list, with a new value assigned to the element at a specified position.

Category

List functions

Function syntax

```
ListSetAt(list, position, value [, delimiters, includeEmptyValues ])
```

See also

ListDeleteAt, ListGetAt, ListInsertAt: Lists in Data types- Developing guide in the Developing ColdFusion Applications

History

ColdFusion MX: Changed delimiter modification: ColdFusion MX does not modify delimiters in the list. (In earlier releases, in some cases, replaced delimiters with the first character in the delimiters parameter.)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>position</td>
<td>A positive integer or a variable that contains one. Position at which to set a value. The first list position is 1.</td>
</tr>
<tr>
<td>value</td>
<td>An element or a list of elements.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

Usage

When assigning an element to a list, ColdFusion inserts a delimiter. If delimiters contains more than one
delimiter, ColdFusion uses the first delimiter in the string, or, if delimiters was omitted, a comma. ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

Example

```coldfusion
<h3>ListSetAt Example</h3>
<!---- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
SELECT Username, Subject, Posted
FROM Messages
</cfquery>

<cfset temp = ValueList(GetMessageUser.Subject)>

<!---- loop through the list and show it with ListGetAt --->
<h3>This is a list of <cfoutput>#ListLen(temp)#</cfoutput> subjects posted in messages.</h3>

<cfset ChangedElement = ListGetAt(temp, 2)>
<cfset TempToo = ListSetAt(temp, 2, "I changed this subject", ",","")>
<ul>
<cfloop From = "1" To = "#ListLen(TempToo)#" INDEX = "Counter">
<cfoutput><li>(#Counter#) SUBJECT: #ListGetAt(TempToo, Counter)#</cfoutput>
</cfloop>
</ul>
<p>Note that element 2, "<cfoutput>#ChangedElement#</cfoutput>", has been altered to "I changed this subject" using ListSetAt.
**ListSort**

**Description**
Sorts list elements according to a sort type and sort order.

**Returns**
A copy of a list, sorted.

**Category**
List functions

**Function syntax**

```
ListSort(list, sort_type [, sort_order, delimiters, includeEmptyValues ])
```

**See also**
Lists in Using ColdFusion Variables in the Developing ColdFusion Applications

**History**
ColdFusion 10: Added support for all Java supported locale-specific characters (including support for umlaut characters). A flag for this support has been added for sorttype = "text" or sorttype = "textnocase".

ColdFusion MX: Changed the order in which sorted elements are returned: in a textnocase, descending sort, this function might return elements in a different sort order than in earlier releases. If sort_type = "textnocase" and sort_order = "desc", ColdFusion MX processes elements that differ only in case differently from earlier releases. ColdFusion MX outputs the elements in the reverse of the ascending order. Earlier releases do not change order of elements that differ only in case. Both operations are correct. The new operation ensures that an ascending and descending sort output elements in exactly reverse order. For example, in a textnocase, desc sort of d, a, a, b, A, the following occurs:

- ColdFusion MX returns d, b, A, a, a
- Earlier ColdFusion releases return d, b, a, a, A

In a textnocase, asc sort, all ColdFusion releases return a, a, A, b, d.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeEmptyValues</td>
<td>Optional. Set to yes to include empty values.</td>
</tr>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>localeSensitive</td>
<td>Specify if you wish to do a locale sensitive sorting.</td>
</tr>
<tr>
<td></td>
<td>The default value is false.</td>
</tr>
</tbody>
</table>
| sort_type          | • numeric: sorts numbers  
|                   | • text: sorts text alphabetically, taking case into 
|                   | account (also known as case sensitive). All letters of 
|                   | one case precede the first letter of the other case:  
|                   | • aabzABZ, if sort_order = "asc" (ascending sort)- ZBAzbaa, if sort_order = "desc" (descending sort)  
|                   | • textnocase: sorts text alphabetically, without regard 
|                   | to case (also known as case-insensitive). A letter in 
|                   | varying cases precedes the next letter:  
|                   | • aAaBbBzzZ, in an ascending sort; preserves original 
|                   | intra-letter order - ZzzBbBaAa, in a descending sort; 
|                   | reverses original intra-letter order |

| sort_order        | • asc - ascending sort order. Default.  
|                   | • aabzABZ or aAaBbBzzZ, depending on value of sort_type, for letters- from smaller to larger, for numbers  
|                   | • desc - descending sort order.  
|                   | • ZBAzbaa or ZzzBbBaAa, depending on value of sort_type, for letters- from larger to smaller, for numbers |

| delimiters        | A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion uses the first character in the string as the delimiter, and ignores the rest. |

**Usage**

ColdFusion ignores empty list elements; thus, the list "a,b,c,,,d" has four elements.

**Example**
<h3>ListSort Example</h3>

<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
SELECT   Username, Subject, Posted
FROM     Messages
</cfquery>

<cfset myList = ValueList(GetMessageUser.UserName)>
<p>Here is the unsorted list. </p>
<cfoutput>#myList#</cfoutput>
<p>Here is the list sorted alphabetically:</p>
<cfset sortedList = ListSort(myList, "Text")>
<cfoutput>#sortedList#</cfoutput>

<p>Here is a numeric list that is to be sorted in descending order.</p>
<cfset sortedNums = ListSort("12,23,107,19,1,65","Numeric", "Desc")>
<cfoutput>#sortedNums#</cfoutput>

<p>Here is a list that must be sorted numerically, since it contains
negative and positive numbers, and decimal numbers. </p>
<cfset sortedNums2 = ListSort("23.75;-34,471:100,-9745","Numeric", "ASC", ";,:")>
<cfoutput>#sortedNums2#</cfoutput>

<p>Here is a list to be sorted alphabetically without consideration of case.</p>
<cfset sortedMix = ListSort("hello;123,HELLO:jeans,-345,887;ColdFusion:coldfusion",
   "TextNoCase", "ASC", ";,:")>
<cfoutput>#sortedMix#</cfoutput>
ListToArray

Description

Copies the elements of a list to an array.

Returns

An array

Category

Array functions, Conversion functions, List functions

Function syntax

ListToArray(list [, delimiters[, includeEmptyFields[, multiCharacterDelimiter]]])

See also

ArrayToList: Using Arrays and Structures in the Developing ColdFusion Applications

History

ColdFusion 9: Added the multiCharacterDelimiter parameter.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>A list or a variable that contains one. You define a list variable with a cfset statement.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. ColdFusion treats each character in the string as a delimiter. The default value is comma.</td>
</tr>
<tr>
<td>includeEmptyFields</td>
<td>A Boolean value specifying whether to create empty array entries if there are two delimiters in a row.</td>
</tr>
<tr>
<td></td>
<td>• false-(Default) ignore empty elements in a list; for example, convert a,,c into an array with only two elements.</td>
</tr>
<tr>
<td></td>
<td>• true-Convert empty elements in a list to empty array entries; for example, convert a,,c into an array with three elements, the second of which is empty.</td>
</tr>
</tbody>
</table>
multiCharacterDelimiter | A Boolean value specifying whether the delimiters parameter specifies a multi-character delimiter. The default is false. If this parameter is true, the delimiters parameter must specify a single delimiter consisting of multiple characters. This parameter enables the ListToArray function to convert a list such as the following to an array of color names: `red:|orange:|yellow:|green:|blue:|indigo:|violet`.

**Usage**

ColdFusion, by default, ignores empty list elements; thus, the list "a,b,c,,d" has four elements. ColdFusion treats each character in the delimiters parameter as a separate delimiter. Therefore, if the parameter is ",+" ColdFusion will break the list at either a comma or a plus sign. If you specify a multiCharacterDelimiter parameter, all list elements must be separated by exactly the specified characters. For example, the following code creates an array with three entries, "red, orange", "yellow, green" and "blue, violet".

```cfml
<cfset list = "red,orange,&yellow,green,&blue,violet">
<cfset arr = listToArray (list, ",&",false,true)>
```

**Example**

```cfml
<h3>ListToArray Example</h3>
<!--- Find a list of users who wrote messages --->
<cfquery name = "GetMessageUser" datasource = "cfdocexamples">
SELECT Username, Subject, Posted
FROM Messages
</cfquery>
<cfset myList = ValueList(GetMessageUser.UserName)>
<p>My list is a list with <cfoutput>#ListLen(myList)#</cfoutput> elements.
<cfset myArrayList = ListToArray(myList, ",",true,false)>
<p>My array list is an array with <cfoutput>#ArrayLen(myArrayList)#</cfoutput> elements.
```
**ListValueCount**

**Description**

Counts instances of a specified value in a list. The search is case sensitive.

**Returns**

The number of instances of `value` in the list.

**Category**

List functions, String functions

**Function syntax**

```
ListValueCount(list, value [, delimiters ])
```

**See also**

[ListValueCountNoCase]; Lists in Data types- Developing guide in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>value</td>
<td>String or number, or a variable that contains one. Item for which to search. The search is case sensitive.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one character, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

**Example**

```
<cfquery name="SearchByDepartment" datasource="cfdocexamples">
SELECT Department
FROM Employees
</cfquery>

<h3>ListValueCount Example</h3>
<p>This example uses ListValueCount to count employees in a department.</p>

<form action="listvaluecount.cfm">
<p>Select a department:</p>
<select name="departmentName">
<option value="Accounting">Accounting</option>
<option value="Administration">Administration</option>
<option value="Engineering">Engineering</option>
<option value="Sales">Sales</option>
</select>
<input type="Submit" name="Submit" value="Search Employee List">
</form>

<!--- wait to have a string for searching defined --->
<cfif IsDefined("FORM.Submit") and IsDefined("FORM.departmentName")>
<cfsset myList = ValueList(SearchByDepartment.Department)>
<cfsset numberInDepartment = ListValueCount(myList, FORM.departmentName)>

<cfsif numberInDepartment is 0>
<h3>There are no employees in #FORM.departmentName#</h3>
</cfsif>
<cfselseIf numberInDepartment is 1>
<p>There is only one person in #FORM.departmentName#.</p>
</cfselseIf>
<cfselse>
<p>There are #numberInDepartment# people in #FORM.departmentName#.</p>
</cfselseIf>
</cfif>
**ListValueCountNoCase**

**Description**

Counts instances of a specified value in a list. The search is case-insensitive.

**Returns**

The number of instances of `value` in the list.

**Category**

List functions

**Function syntax**

```
ListValueCountNoCase(list, value [, delimiters ])
```

**See also**

[ListValueCount]: Lists in Data types- Developing guide in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>A list or a variable that contains one.</td>
</tr>
<tr>
<td>value</td>
<td>String or number, or a variable that contains one. Item for which to search. The search is case-insensitive.</td>
</tr>
<tr>
<td>delimiters</td>
<td>A string or a variable that contains one. Characters that separate list elements. The default value is comma. If this parameter contains more than one value, ColdFusion processes each occurrence of each character as a delimiter.</td>
</tr>
</tbody>
</table>

**Example**
<cfquery name = "SearchByDepartment" datasource = "cfdocexamples">
SELECT  Department
FROM  Employees
</cfquery>

<h3>ListValueCountNoCase Example</h3>
<p>This example uses ListValueCountNoCase to count employees in a department.</p>

<form action = "listvaluecountnocase.cfm">
<p>Select a department:</p>
<select name = "departmentName">
<option value = "Accounting">Accounting</option>
<option value = "Administration">Administration</option>
<option value = "Engineering">Engineering</option>
<option value = "Sales">Sales</option>
</select>
<input type = "Submit" name = "Submit" value = "Search Employee List">
</form>

<!--- wait to have a string for searching defined --->
<cfif IsDefined("FORM.Submit") and IsDefined("FORM.departmentName")>
<cfset myList = ValueList(SearchByDepartment.Department)>
<cfset numberInDepartment = ListValueCountNoCase(myList, FORM.departmentName)>

<cfif numberInDepartment is 0>
<h3>There are no employees in #FORM.departmentName#</h3>
</cfif>
<cfelseIf numberInDepartment is 1>
<p>There is only one person in #FORM.departmentName#.</p>
</cfelse>
<cfelse>
<p>There are #numberInDepartment# people in #FORM.departmentName#.</p>
</cfelse>
</cfif>
LJustify

Description

Left justifies characters in a string of a specified length.

Returns

A copy of a string, left-justified.

Category

Display and formatting functions, String functions

Function syntax

LJustify(string, length)

See also

CJustify, RJustify

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
<tr>
<td>length</td>
<td>Length of field in which to justify string</td>
</tr>
</tbody>
</table>

Example
<!--- This example shows how to use LJustify --->
<cfparam name = "jstring" default = "">

<cfif IsDefined("FORM.justifyString")>
  <cfset jstring = LJustify(FORM.justifyString, 35)>
</cfif>
<html>
<head>
  <title>LJustify Example</title>
</head>
<body>
<h3>LJustify Function</h3>
<p>Enter a string, and it will be left justified within the sample field</p>
<form action = "ljustify.cfm">
  <p><input type = "Text" value = "<cfoutput>#jString#</cfoutput>" size = 35 name = "justifyString"></p>
  <p><input type = "Submit" name = ""> <input type = "RESET"></p>
</form>
Location

Description

A function equivalent of the `cflocation` tag and is used in the `<cfscript>` mode.

Parameters

Same as the `<cflocation>` tag.

Category

Data output functions

Function syntax

```
location("url", addtoken, statusCode);
```

For positional notations, the sequence must be followed exactly in the same manner as provided in the syntax. If you do not provide one of the parameters, use an empty string instead. This does not apply to Boolean values for which you must provide proper values even if you have to skip them.

See also

cfscript, cflocation

Usage

You can call this function as name=value pair or as positional argument.

Example

```
<cfscript
  location(url="http://localhost:8500/administrator")
</cfscript>
```
Log

Description

 Calculates the natural logarithm of a number. Natural logarithms are based on the constant e (2.71828182845904).

Returns

 The natural logarithm of a number.

Category

 Mathematical functions

Function syntax

```
Log(number)
```

See also

 Exp, Log10

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Positive real number for which to calculate the natural logarithm</td>
</tr>
</tbody>
</table>

Example
<h3>Log Example</h3>

<cfif IsDefined("FORM.number")>
  <cfoutput>
    <p>Your number, #FORM.number#</p>
    <br>#FORM.number# raised to the E power: #exp(FORM.number)#
    <cfif FORM.number LTE 0><br>Enter a positive real number to get its natural logarithm</cfif>
    <cfelse><br>The natural logarithm of #FORM.number#: #log(FORM.number)#</cfif>
  </cfoutput>
  <cfform action = "log.cfm">
    <cfinput type = "Text" name = "number" message = "You must enter a number" validate = "float" required = "No">
    <input type = "Submit" name = "">
  </cfform>
</cfif>
<cfif FORM.number LTE 0><br>Enter a positive real number to get its logarithm to base 10</cfif>
  <cfelse><br>The logarithm of #FORM.number# to base 10: #log10(FORM.number)#</cfif>
Log10

Description

Calculates the logarithm of number, to base 10.

Returns

Number; the logarithm of number, to base 10.

Category

Mathematical functions

Function syntax

```
Log10(number)
```

See also

Exp, Log

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Positive real number for which to calculate the logarithm</td>
</tr>
</tbody>
</table>

Example
<h3>Log10 Example</h3>
<cfif IsDefined("FORM.number")>
<coutput>
<p>Your number, #FORM.number#</p>
<br>#FORM.number# raised to the E power: #exp(FORM.number)#
<br>You must enter a positive real number to see the natural logarithm of that number</br>
</cfif>
</coutput>
</cfif>

<cfform action = "log10.cfm">
Enter a number to find its value raised to the E power, its natural logarithm, and the logarithm of number to base 10.
<cfinput type = "Text" name = "number" message = "You must enter a number" validate = "float" required = "No">
<input type = "Submit" name = ""/>
</cfform>
LSCurrencyFormat

Description

Formats a number in a locale-specific currency format. For countries that use the euro, the result depends on the JVM.

Returns

A formatted currency value.

Category

Display and formatting functions, International functions

Function syntax

LSCurrencyFormat(number [, type, locale])

See also

LSEuroCurrencyFormat, LSIsCurrency, LSParseCurrency, LSParseEuroCurrency, SetLocale; Handling data in ColdFusion in the Developing ColdFusion Applications

History

ColdFusion 8: Added the locale parameter.

ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. If a negative number is passed to it, it returns a negative number. If type = "local", it returns the value in the current locale's standard format. If type = "international", it returns the value in the current locale's international standard format. This function uses Java standard locale formatting rules on all platforms.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Currency value</td>
</tr>
</tbody>
</table>
| type      | • local: the currency format and currency symbol used in the locale.  
            • With JDK 1.3, the default for Euro Zone countries is their local currency.  
            • With JDK 1.4, the default for Euro Zone countries is the euro.  
            • international: the international standard currency format and currency symbol of the locale.  
            • none: the currency format used in the locale; no currency symbol |
<table>
<thead>
<tr>
<th>Locale</th>
<th>Type = Local</th>
<th>Type = International</th>
<th>Type = None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese (China)</td>
<td>100,000.00</td>
<td>CNY100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>Chinese (Hong Kong)</td>
<td>HK$100,000.00</td>
<td>HKD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>Chinese (Taiwan)</td>
<td>NT$100,000.00</td>
<td>TWD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>Dutch (Belgian)</td>
<td>100.000,00 - 100.000,00</td>
<td>BEF100.000,00EUR100.000,00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>Dutch (Standard)</td>
<td>100.000,00fl 100.000,00</td>
<td>NLG100.000,00EUR100.000,00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (Australian)</td>
<td>$100,000.00</td>
<td>AUD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (Canadian)</td>
<td>$100,000.00</td>
<td>CAD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (New Zealand)</td>
<td>$100,000.00</td>
<td>NZD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (UK)</td>
<td>100,000.00</td>
<td>GBP100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (US)</td>
<td>$100,000.00</td>
<td>USD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>French (Belgian)</td>
<td>100.000,00 - 100.000,00</td>
<td>EUR100.000,00BEF100.000,00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>French (Canadian)</td>
<td>100 000,00 $</td>
<td>CAD100 000,00</td>
<td>100 000,00</td>
</tr>
<tr>
<td>French (Standard)</td>
<td>100 000,00 - 100 000,00</td>
<td>EUR100 000,00FRF100 000,00</td>
<td>100 000,00</td>
</tr>
</tbody>
</table>

Usage

This function uses Java standard locale formatting rules on all platforms.

⚠️ Note

With a Sun 1.3.1-compliant JVM, use the `LSEuroCurrencyFormat` function to format euro currency values.

Currency output

The following table shows sample currency output. For locales that use Euro, the Local and International columns contains two entries. The first is entry is the result with a Sun the 1.4.1 or later compliant JVM, the second entry is the result with a 1.3.1-compliant JVM.
<table>
<thead>
<tr>
<th>Language</th>
<th>Display</th>
<th>Currency</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>French (Swiss)</td>
<td>SFr. 100'000.00</td>
<td>CHF100'000.00</td>
<td>100'000.00</td>
</tr>
<tr>
<td>German (Austrian)</td>
<td>100.000,00-S</td>
<td>EUR100.000,00ATS100.000,00</td>
<td>100.000,00</td>
</tr>
<tr>
<td>German (Standard)</td>
<td>100.000,00 - 100.000,00 DM</td>
<td>EUR100.000,00DEM100.000,00</td>
<td>100.000,00</td>
</tr>
<tr>
<td>German (Swiss)</td>
<td>SFr. 100'000.00</td>
<td>CHF100'000.00</td>
<td>100'000.00</td>
</tr>
<tr>
<td>Italian (Standard)</td>
<td>100.000,00L. 10.000.000</td>
<td>EUR10.000.000ITL10.000.000</td>
<td>10.000.000</td>
</tr>
<tr>
<td>Italian (Swiss)</td>
<td>SFr. 100'000.00</td>
<td>CHF100'000.00</td>
<td>100'000.00</td>
</tr>
<tr>
<td>Japanese</td>
<td>100,000</td>
<td>JPY100,000</td>
<td>JPY100,000</td>
</tr>
<tr>
<td>Korean</td>
<td>W100,000</td>
<td>KRW100,000</td>
<td>100,000</td>
</tr>
<tr>
<td>Norwegian (Bokmal)</td>
<td>kr 100 000,00</td>
<td>NOK100 000,00</td>
<td>100 000,00</td>
</tr>
<tr>
<td>Norwegian (Nynorsk)</td>
<td>kr 100 000,00</td>
<td>NOK100 000,00</td>
<td>100 000,00</td>
</tr>
<tr>
<td>Portuguese (Brazilian)</td>
<td>R$100.000,00</td>
<td>BRC100.000,00</td>
<td>100.000,00</td>
</tr>
<tr>
<td>Portuguese (Standard)</td>
<td>100.000,00 - R$100.000,00</td>
<td>EUR100.000,00BRC100.000,00</td>
<td>100.000,00</td>
</tr>
<tr>
<td>Spanish (Mexican)</td>
<td>$100,000,00</td>
<td>MXN100,000,00</td>
<td>100,000,00</td>
</tr>
<tr>
<td>Spanish (Modern)</td>
<td>100.000,00 - 10.000.000 Pts</td>
<td>EUR10.000.000ESP10.000.000</td>
<td>10.000.000</td>
</tr>
<tr>
<td>Spanish (Standard)</td>
<td>100.000,00 - 10.000.000 Pts</td>
<td>ESP10.000.000EUR10.000.000</td>
<td>10.000.000</td>
</tr>
<tr>
<td>Swedish</td>
<td>100.000,00 kr</td>
<td>SEK100.000,00</td>
<td>100.000,00</td>
</tr>
</tbody>
</table>

**Note**

ColdFusion maps Spanish (Modern) to the Spanish (Standard) format.

To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

**Example**
<h3>LSCurrencyFormat Example</h3>
<p>LSCurrencyFormat returns a currency value using the locale convention. Default value is "local."
</p>
<!--- loop through list of locales; show currency values for 100,000 units --->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",,">
<cfset oldlocale = SetLocale(locale)>
<cfoutput><p><b><i>#locale#</i></b><br>
Local: #LSCurrencyFormat(100000, "local")#<br>
International: #LSCurrencyFormat(100000, "international")#<br>
None: #LSCurrencyFormat(100000, "none")#<br>
<hr noshade>
</cfoutput>
</cfloop>
**LSDateFormat**

**Description**

Formats the date part of a date/time value in a locale-specific format.

**Returns**

A formatted date/time value. If no mask is specified, the value is formatted according to the locale setting of the client computer.

**Category**

Date and time functions, Display and formatting functions, International functions

**Function syntax**

```
LSDateFormat(date [, mask, locale])
```

See also

LSParseDateTime, LSTimeFormat, DateFormat, SetLocale; Handling data in ColdFusion in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added the locale parameter. ColdFusion MX:

- Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.
- Added support for the following mask parameter options: short, medium, long, and full.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>
mask

Characters that show how ColdFusion displays the date:

- d: Day of month. Digits; no leading zero for single-digit days
- dd: Day of month. Digits; leading zero for single-digit days
- ddd: Day of week, abbreviation
- dddd: Day of week. Full name
- m: Month. Digits; no leading zero for single-digit months
- mm: Month. Digits; leading zero for single-digit months
- mmm: Month. abbreviation (if appropriate)
- mmmm: Month. Full name
- y: Year. Last two digits; no leading zero for years less than 10
- yy: Year. Last two digits; leading zero for years less than 10
- yyyy: Year. Four digits
- gg: Period/era string. Not processed. Reserved for future use

The following conform to Java locale-specific time encoding standards. Their exact formats depend on the locale:

- short: dd, mm, and yy separated by / marks
- medium: text format using mmm, d, and yyyy
- long: text format using mmmm, d, and yyyy
- full: text format using dddd, mmmm, d, and yyyy

The default value is medium. For more information on formats, see LSParseDateTime.

locale

Locale to use instead of the locale of the page when processing the function

Usage

This function uses Java standard locale formatting rules on all platforms. When passing date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object. To calculate a difference between time zones, use the GetTimeZoneInfo function.

Example
<h3>LSDateFormat Example</h3>
<p>LSDateFormat formats the date part of a date/time value using the locale convention.
</p>
<!--- loop through a list of locales; show date values for Now()--->
<cffloop list = "#Server.Coldfusion.SupportedLocales#"
    index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput>
    <p><b><i>#locale#</i></b><br>
    #LSDateFormat(Now(), "mmm-dd-yyyy")#<br>
    #LSDateFormat(Now(), "mmmm d, yyyy")#<br>
    #LSDateFormat(Now(), "mm/dd/yyyy")#<br>
    #LSDateFormat(Now(), "d-mmm-yyyy")#<br>
    #LSDateFormat(Now(), "ddd, mmmm dd, yyyy")#<br>
    #LSDateFormat(Now(), "d/m/yy")#<br>
    #LSDateFormat(Now())#<br>
    </cfoutput>
    <hr noshade>
</cffloop>
**LSDateTimeFormat**

**Description**

Formats date and time values using locale-specific date and time formatting conventions.

**Returns**

A formatted date and time value.

**Category**

*Date and time functions, Display and formatting functions, International functions*

**Syntax**

```
LSDateTimeFormat (date , mask)
LSDateTimeFormat (date [, mask, locale])
LSDateTimeFormat (date [, mask, locale, timeZone])
```

**See also**

[LSDateTimeFormat](#), [LSTimeFormat](#), [DateFormat](#), [SetLocale](#); in the *Handling data in ColdFusion* in the *Developing ColdFusion Applications*

**History**

ColdFusion 10: Added the function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Required. A date/time object, in the range 100 AD-9999 AD.</td>
</tr>
<tr>
<td>mask</td>
<td>Optional. Mask that has to be used for formatting. See <a href="#">DateTimeFormat</a>.</td>
</tr>
</tbody>
</table>
| timeZone  | The time-zone information. You can specify in either of the following formats:  
- Abbreviation such as GMT or PST  
- Full name such as Europe/Dublin  
By default, this is the time-zone followed by the system. |
| locale    | Locale to use instead of the locale of the page when processing the function. |

**Example**
<cfset todayDateTime = Now()>
<body>
<h3>DateAndTimeFormat Example</h3>
<p>Today's date and time are <cfoutput>#todayDateTime#</cfoutput>.</p>
<p>Using DateTimeFormat, we can display that date and time in different ways:
<cfoutput>
<ul>
<li>#DateTimeFormat(todayDateTime, "yyyy.MM.dd G 'at' HH:nn:ss z")#</li>
<li>#DateTimeFormat(todayDateTime, "EEE, MMM d, 'yy")#</li>
<li>#DateTimeFormat(todayDateTime, "h:nn a")#</li>
<li>#DateTimeFormat(todayDateTime, "hh 'o''clock' a, zzzz")#</li>
<li>#DateTimeFormat(todayDateTime, "K:nn a, z")#</li>
<li>#DateTimeFormat(todayDateTime, "yyyy.MM.dd MMMM d GGG hh:nn aaa")#</li>
<li>#DateTimeFormat(todayDateTime, "EEE, d MMM yyyy HH:nn:ss Z")#</li>
<li>#DateTimeFormat(todayDateTime, "yyMMddHHnnssZ", "English (UK)", "GMT")#</li>
</ul>
</cfoutput>
LSEuroCurrencyFormat

Description

Formats a number in a locale-specific currency format.

Returns

A formatted currency value. For countries in the Euro currency zone, the function uses the locale's rule's for formatting currency in euros.

Category

Display and formatting functions, International functions

Function syntax

LSEuroCurrencyFormat(currency-number [, type, locale])

See also

LSParseEuroCurrency, LSCurrencyFormat, SetLocale; Locale-specific content in the Developing ColdFusion Applications

History

ColdFusion 8: Added the locale parameter.

ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java locale formatting rules on all platforms, except that it uses the rule detailed in the Usage section for countries in the Euro currency zone. As a result, it format currencies for non-Euro zone locales using the country's currency, not euros.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>currency-number</td>
<td>Currency value.</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
<tr>
<td>type</td>
<td>• local: the currency format used in the locale. (Default.)</td>
</tr>
<tr>
<td></td>
<td>• international: the international standard currency format of the locale. For example, EUR10.00</td>
</tr>
<tr>
<td></td>
<td>• none: the currency format used in the locale; no currency symbol</td>
</tr>
</tbody>
</table>

Usage

This function uses euro currency formatting rules for all JVM versions, as follows:
If the country of the current locale belongs to the Euro Zone (whose members have converted to the euro) the formatted output for the local type includes the Euro currency sign; for the international type, the output includes the euro currency symbol (EUR). If the value is negative, the format includes a negative sign before the value or parentheses around the value, according to the formatting rules of the current locale.

If the country of the current locale is not in the Euro Zone, the currency sign or symbol of the current locale displays. If the value is negative, the format includes a negative sign before the value or parentheses around the value, according to the formatting rules of the current locale. For a list of the locale options that ColdFusion supports, and information on setting the default display format of date, time, number, and currency values, see SetLocale.

### Currency output

The following table shows examples of currency output:

<table>
<thead>
<tr>
<th>Locale</th>
<th>Type = Local</th>
<th>Type = International</th>
<th>Type = None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese (China)</td>
<td>100,000.00</td>
<td>CNY100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>Chinese (Hong Kong)</td>
<td>HK$100,000.00</td>
<td>HKD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>Chinese (Taiwan)</td>
<td>NT$100,000.00</td>
<td>TWD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>Dutch (Belgian)</td>
<td>100,000.00</td>
<td>EUR100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>Dutch (Standard)</td>
<td>100,000.00</td>
<td>EUR100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (Australian)</td>
<td>$100,000.00</td>
<td>AUD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (Canadian)</td>
<td>$100,000.00</td>
<td>CAD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (New Zealand)</td>
<td>$100,000.00</td>
<td>NZD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (UK)</td>
<td>100,000.00</td>
<td>GBP100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>English (US)</td>
<td>$100,000.00</td>
<td>USD100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>French (Belgian)</td>
<td>100,000.00</td>
<td>EUR100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>French (Canadian)</td>
<td>100 000.00 $</td>
<td>CAD100 000.00</td>
<td>100 000.00</td>
</tr>
<tr>
<td>French (Standard)</td>
<td>100 000.00</td>
<td>EUR100 000.00</td>
<td>100 000.00</td>
</tr>
<tr>
<td>French (Swiss)</td>
<td>SFr. 100'000.00</td>
<td>CHF100'000.00</td>
<td>100'000.00</td>
</tr>
<tr>
<td>German (Austrian)</td>
<td>100,000.00</td>
<td>EUR100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>German (Standard)</td>
<td>100,000.00</td>
<td>EUR100,000.00</td>
<td>100,000.00</td>
</tr>
<tr>
<td>German (Swiss)</td>
<td>SFr. 100'000.00</td>
<td>CHF100'000.00</td>
<td>100'000.00</td>
</tr>
</tbody>
</table>
### Note

ColdFusion uses the Spanish (Standard) formats for Spanish (Modern) and Spanish (Standard).

The following example shows how the function formats negative values. The format includes a negative sign before the value, or parentheses around the value, according to the formatting rules of the current locale.

<table>
<thead>
<tr>
<th>Input value</th>
<th>Output if locale = French (Standard)</th>
<th>Output if locale = English (US)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1234.56</td>
<td>-1 234,56</td>
<td>($1,234.56)</td>
</tr>
</tbody>
</table>

**Example**
<h3>LSEuroCurrencyFormat Example</h3>
<p>LSEuroCurrencyFormat returns a currency value using the locale convention. Default value is "local."
</p> <!-- Loop through list of locales, show currency values for 100,000 units -->
<cfloop list = "#Server.Coldfusion.SupportedLocales#"
    index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p><B><I>#locale#</I></B><br>
    Local: #LSEuroCurrencyFormat(100000, "local")#<br>
    International: #LSEuroCurrencyFormat(100000, "international")#<br>
    None: #LSEuroCurrencyFormat(100000, "none")#<br>
    <hr noshade>
    </cfoutput>
</cfloop>
**LSIsCurrency**

**Description**
Determines whether a string is a valid representation of a currency amount in the current locale.

**Returns**
True, if the parameter is formatted as a valid currency amount, including the appropriate currency indicator. The return value is True for amounts in the local, international, or none currency formats.

**Category**
Display and formatting functions, Decision functions, International functions

**Function syntax**

```
LSIsCurrency(string [, locale])
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A currency string or a variable that contains one.</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>

**History**
ColdFusion 8: Added the locale parameter.

ColdFusion MX: Changed formatting behavior: this function might return a different result than in earlier releases. This function uses Java standard locale formatting rules on all platforms; the results might vary depending upon the JVM; for example, Sun JVM 1.4.1 requires euro format the local currency if the current locale’s country belongs to the Euro Zone.

**Usage**
For examples of ColdFusion code and output that shows differences between earlier ColdFusion releases and ColdFusion MX in accepting input formats and displaying output, see [LSCurrencyFormat](#).
Note

If the locale belongs to a Euro zone country and the currency is a correctly formatted euro value for the locale, this function returns True for all JVMs, including Sun 1.3.1. As a result, with 1.3.1-compliant JVMs, the LSIscurrency function does not ensure that LSParseCurrency returns a value. If a currency uses the older country-specific format for Euro Zone locales, the LSIscurrency function returns False for newer JVMs, such as Sun 1.4.1 and 1.6, and True for older JVMs, such as Sun 1.3.1.

Note

To set the default display format of date, time, number, and currency values, use the SetLocale function.

Example

```html
<h3>LSIsCurrency Example</h3>

<cfif IsDefined("FORM.locale")>
!--- if locale is defined, set locale to that entry --->
<cfset NewLocale = SetLocale(FORM.locale)>

<p>Is the value 
<coutput>#FORM.myValue#</coutput>

a proper currency value for 
<coutput>#GetLocale()#</coutput>?

Answer: 
<coutput>#LSIsCurrency(FORM.myValue)#</coutput>
</cfif>

<p><form action = "LSIsCurrency.cfm">
Select a locale for which you would like to check a currency value:
<cf--- check the current locale for server --->
<cfset serverLocale = GetLocale()>
```
**LSIsDate**

**Description**

Determines whether a string is a valid representation of a date/time value in the current locale.

**Returns**

True, if the string can be formatted as a date/time value in the current locale; False, otherwise.

**Category**

Date and time functions, Display and formatting functions, International functions

**Function syntax**

```
LSIsDate(string [, locale])
```

**See also**

CreateDateTime, GetLocale, IsNumericDate, LSDateFormat, ParseDateTime, SetLocale; Handling data in ColdFusion in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added the locale parameter.

- Changed formatting behavior: this function might return a different result than in earlier releases. This function uses Java standard locale formatting rules on all platforms.
- Changed behavior: this function accepts a dash or hyphen character only in the Dutch(Standard) and Portuguese (Standard) locales. If called this way (for example, LsIsDate("3-1-2002") in any other locale, this function returns False. (Earlier releases returned True.)
- Changed behavior: when using the SUN JRE 1.3.1 on an English(UK) locale, this function returns False for a date that has a one-digit month or day (for example, 1/1/01). To work around this, insert a zero in a one-digit month or day (for example, 01/01/01).

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>

**Usage**

A date/time object is in the range 100 AD-9999 AD. To set the default display format of date, time, number, and currency values, use the SetLocale function.

**Example**
<h3>LSIsDate Example</h3>
<cfif IsDefined("FORM.locale")>
<!---- if locale is defined, set locale to that entry --->
<cfset NewLocale = SetLocale(FORM.locale)>
<p>Is the value "<cfoutput>#FORM.myValue#</cfoutput>" a proper date value for <cfoutput>#GetLocale()#</cfoutput>?<br>
<p>Answer: <cfoutput>#LSIsDate(FORM.myValue)#</cfoutput><br>
</cfif>
<p><form action = "LSIsDate.cfm">
<p>Select a locale for which you would like to check a date value:<br>
<!---- check the current locale for server --->
<cfset serverLocale = GetLocale()>

**LSIsNumeric**

**Description**

Determines whether a string is a valid representation of a number in the current locale.

**Returns**

True, if the string represents a number the current locale; False, otherwise.

**Category**

Decision functions, International functions, String functions

**Function syntax**

```
LSIsNumeric(string [, locale])
```

**See also**

GetLocale, SetLocale: Handling data in ColdFusion in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX: Changed formatting behavior: this function might return a different result than in earlier releases. This function uses Java standard locale formatting rules on all platforms.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>

**Usage**

To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

**Example**
<h3>LSIsNumeric Example</h3>

<cfif IsDefined("FORM.locale")>
<!--- if locale is defined, set locale to that entry --->
<cfset NewLocale = SetLocale(FORM.locale)>

<p>Is the value "<cfoutput>#FORM.myValue#</cFOUTPUT>" a proper numeric value for <cfoutput>#GetLocale()#</cfoutput>?

<p>Answer: <cfoutput>#LSIsNumeric(FORM.myValue)#</cfoutput>
</cfif>

<p><form action = "LSIsNumeric.cfm">
<br><p>Select a locale for which to check a numeric value:
...
LSNumberFormat

Description

Formats a number in a locale-specific format.

Returns

A formatted number.

- If no mask is specified, it returns the number formatted as an integer
- If no mask is specified, truncates the decimal part; for example, it truncates 34.57 to 35
- If the specified mask cannot correctly mask a number, it returns the number unchanged
- If the parameter value is "" (an empty string), it returns 0.

Category

Display and formatting functions, International functions

Function syntax

```
LSNumberFormat(number [, mask, locale])
```

See also

GetLocale, SetLocale; Handling data in ColdFusion in the Developing ColdFusion Applications

History

ColdFusion 8: Added the locale parameter.

ColdFusion MX:

- Changed behavior: if the specified mask format cannot correctly mask a number, this function returns the number unchanged. (In earlier releases, it truncated the number or threw an error.) (If no mask is specified, ColdFusion MX truncates the decimal part as ColdFusion 5 does. For example, it truncates 1234.567 to 1235.)
- Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Number to format</td>
</tr>
<tr>
<td>mask</td>
<td>LSNumberFormat mask characters apply, except: dollar sign, comma, and dot are mapped to their locale-specific equivalents.</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>
The following table lists the **LSNumberFormat** mask characters:

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>(Underscore.) Digit placeholder.</td>
</tr>
<tr>
<td>9</td>
<td>Digit placeholder. (Shows decimal places more clearly than _. )</td>
</tr>
<tr>
<td>.</td>
<td>Location of a mandatory decimal point (or locale-appropriate symbol).</td>
</tr>
<tr>
<td>0</td>
<td>Located to the left or right of a mandatory decimal point. Pads with zeros.</td>
</tr>
<tr>
<td>()</td>
<td>If number is less than zero, puts parentheses around the mask.</td>
</tr>
<tr>
<td>+</td>
<td>Puts plus sign before positive number; minus sign before negative number.</td>
</tr>
<tr>
<td>-</td>
<td>Puts space before positive number; minus sign before negative number.</td>
</tr>
<tr>
<td>,</td>
<td>Separates every third decimal place with a comma (or locale-appropriate symbol).</td>
</tr>
<tr>
<td>L,C</td>
<td>Left-justifies or center-justifies number within width of mask column. First character of mask must be L or C. The default value is right-justified.</td>
</tr>
<tr>
<td>$</td>
<td>Puts a dollar sign (or locale-appropriate symbol) before formatted number. First character of mask must be the dollar sign ($).</td>
</tr>
<tr>
<td>^</td>
<td>Separates left and right formatting.</td>
</tr>
</tbody>
</table>

**Note**

If you do not specify a sign for the mask, positive and negative numbers do not align in columns. To put a plus sign or space before positive numbers and a minus sign before negative numbers, use the plus or hyphen mask character, respectively.

**Usage**

This function uses Java standard locale formatting rules on all platforms. The position of symbols in format masks determines where the codes take effect. For example, if you put a dollar sign at the far left of a format mask, ColdFusion displays a dollar sign at the left edge of the formatted number. If you separate the dollar sign on the left edge of the format mask by at least one underscore, ColdFusion displays the dollar sign just to the left of the digits in the formatted number. These examples show how symbols determine formats:
The positioning can also show where to put a minus sign for negative numbers:

<table>
<thead>
<tr>
<th>Number</th>
<th>Mask</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4.37</td>
<td>-__<em>.</em></td>
<td>&quot;- 4.37&quot;</td>
</tr>
<tr>
<td>-4.37</td>
<td><em>.</em></td>
<td>&quot; -4.37&quot;</td>
</tr>
</tbody>
</table>

The positions for a symbol are: far left, near left, near right, and far right. The left and right positions are determined by the side of the decimal point on which the code character is shown. For formats that do not have a fixed number of decimal places, you can use a caret (^) to separate the left fields from the right. An underscore determines whether the code is placed in the far or near position. Most code characters effect is determined by the field in which they are located. This example shows how to specify where to put parentheses to display negative numbers:

<table>
<thead>
<tr>
<th>Number</th>
<th>Mask</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.21</td>
<td>C(<em>^</em>)</td>
<td>&quot;( 3.21 )&quot;</td>
</tr>
<tr>
<td>3.21</td>
<td>C_(<em>^</em>)</td>
<td>&quot; (3.21 )&quot;</td>
</tr>
<tr>
<td>3.21</td>
<td>C(_^)</td>
<td>&quot;( 3.21 )&quot;</td>
</tr>
<tr>
<td>3.21</td>
<td>C_(_^)</td>
<td>&quot; (3.21 )&quot;</td>
</tr>
</tbody>
</table>

To set the default display format of date, time, number, and currency values, use the `SetLocale` function. When converting from string to double, to prevent rounding errors, this function adds a rounding factor of 1.5543122344752E-014 to the converted number. For example, without adding the rounding factor, converting the string value 1.275 to double with two digits of precision results in a value of 1.27499999999999999, which would be rounded up to 1.27. By adding the rounding factor, the conversion correctly results in a value of 1.28. If you round off a double, such as 1.99499999999999999999999999999, where the last decimal is 10E-14, the rounding factor can cause an incorrect result.

Example
<h3>LSNumberFormat Example</h3>
<p>LSNumberFormat returns a number value using the locale convention.

<!--- loop through a list of locales and show number values --->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
    index = "locale" delimiters = ",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p><b><i>#locale#</i></b><br>
    #LSNumberFormat(-1234.5678, "_________")#<br>
    #LSNumberFormat(-1234.5678, "_________.___")#<br>
    #LSNumberFormat(1234.5678, "_________")#<br>
    #LSNumberFormat(1234.5678, "_________.___")#<br>
    #LSNumberFormat(1234.5678, "$(_________.___)")#<br>
    #LSNumberFormat(-1234.5678, "$(_________.___)")#<br>
    #LSNumberFormat(1234.5678, "+_________.___")#<br>
    #LSNumberFormat(-1234.5678, "-_________.___")#<br>
</cfoutput>
</cfloop>
**LSParseCurrency**

**Description**

Converts a locale-specific currency string into a formatted number. Attempts conversion by comparing the string with each of the three supported currency formats (none, local, international) and using the first that matches.

**Returns**

A formatted number (string representation of a number) that matches the value of the parameter.

**Category**

International functions, String functions

**Function syntax**

```
LSParseCurrency(string [, locale])
```

**See also**

LSParseEuroCurrency, LSCurrencyFormat, LSEuroCurrencyFormat, LSIsCurrency; Locale-specific content in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added the `locale` parameter.

ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A locale-specific string a variable that contains one</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>

**Usage**

This function uses the locale formatting rules of the JVM (specified in the ColdFusion Administrator Java and JVM page) on all platforms. These rules changed between Sun JVM 1.3.1 and JVM 1.4.1:

- JVM 1.3.1 requires that the local and international versions of currencies of countries in the Euro zone be formatted using the older, country-specific designations, such as 100.000,00 DM or DEM100.000,00 for the German (Standard) locale. Use the LSParseEuroCurrency function to parse euro currencies in these locales with JVM 1.3.1.
- JVM 1.4.1 requires that currencies for Euro zone countries be expressed as euros; for example 100.000,00 BADCHAR or EUR100.000,00.
Note

The `LSIsCurrency` function always returns True if the locale is in the Euro currency zone and the currency is expressed in euros, including when using JVM 1.3.1. As a result, with older JVMs, `LSIsCurrency` does not ensure that `LSParseCurrency` returns a value.

To set the default display format of date, time, number, and currency values, use the `SetLocale` function. For a list of the locale-specific formats used to parse the currency, see `LSCurrencyFormat`.

Example

```
<h3>LSParseCurrency Example</h3>
<p>LSParseCurrency coverts a locale-specific currency string to a number. Attempts conversion through each of the three default currency formats. <!--- loop through a list of locales; show currency values for 123,456 units --->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
    index = "locale" delimiters = ",,">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p><B><I>#locale#</I></B><br>
        Local: #LSCurrencyFormat(123456.78, "local")#<br>
        Parsed local Currency: #LSParseCurrency(LSCurrencyFormat(123456,"local"))#<br>
        International: #LSCurrencyFormat(123456.78999, "international")#<br>
        Parsed International Currency: #LSParseCurrency(LSCurrencyFormat(123456.78999,"international"))#<br>
        None: #LSCurrencyFormat(123456.78999, "none")#<br>
        Parsed None formatted currency: #LSParseCurrency(LSCurrencyFormat(123456.78999,"none"))#<br>
    </cfoutput>
</cfloop>
```
**LSParseDateTime**

**Description**

Converts a string that is a valid date/time representation in the current locale into a date/time object.

**Returns**

A date/time object.

**Category**

*Date and time functions, Display and formatting functions, International functions, String functions*

**Function syntax**

```
LSParseDateTime(date/time-string [, locale, format])
```

**See also**

*LSDateFormat, ParseDateTime, SetLocale, GetLocale; Locales* in the *Developing ColdFusion Applications*

**History**

ColdFusion 10: Added the format parameter.

ColdFusion 8: Added the locale parameter.

ColdFusion MX:

- Changed formatting behavior: this function might not parse string formats that worked with earlier releases. This function uses Java standard locale formatting rules on all platforms.
- Changed how the date/time-string parameter value is processed: ColdFusion processes the date/tim e-string parameter value time zone information differently than in earlier releases, as described in the Usage section.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date/time-string</td>
<td>A string a variable that contains one, in a format that is readable in the current locale.</td>
</tr>
<tr>
<td>format</td>
<td>Optional, provides the format of the string. This string is used to parse the given date string to date time object.</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>

**Usage**

This function can parse any date, time, or date/time combination that conforms to Java standard locale formatting rules for the current locale. The following table lists some of the date/time values you can pass to this function in the English (US) locale. You can also pass only the date or the time parts of these formats:
Valid dates are in the range 100 AD-9999 AD. Two-digit years in the range 00-29 are interpreted as being 2000-2029. Two-digit years in the range 30-99 are interpreted as being 1930-1999. This function corrects for differences between the current time zone and any time zone specified in the input parameter.

- If a time zone specified in the `date/time-string` parameter is different from the time zone setting of the computer, ColdFusion adjusts the time value to its equivalent in the computer time zone.
- If a time zone is not specified in the `date/time-string` parameter, ColdFusion does not adjust the time value.

### Note
This function does not accept POP dates, which include a time zone offset value.

In the following example, the parameter `format` specifies the way in which the given date string should be read:

```cfc
<cfoutput>lsParseDateTime("01/08/2011","en","MM/dd/yyyy")</cfoutput>
```

---

**Format** | **Example**
--- | ---
m/dd/yy h:mm:ss | 1/30/02 7:02:33
m/dd/yy h:mm tt | 1/30/02 7:02 AM
m/dd/yyyy h:mm | 1/30/2002 7:02 AM
mmm dd, yyyy h:mm:ss tt | Jan 30, 2002 7:02:12 AM
mmmm dd, yyyy h:mm:ss tt zzz | January 30, 2002 7:02:23 AM PST
ddd, mmm dd, yyyy hh:mm:ss | Wed, Jan 30, 2002 07:02:12
dddd, mmmm dd, yyyy h:mm:ss tt zzz | Wednesday, January 30, 2002 7:02:12 AM PST
<h3>LSParseDateTime Example - returns a locale-specific date/time object</h3>

<!--- loop through a list of locales and show date values for Now()--->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",">
  <cfset oldlocale = SetLocale(locale)>
  <cfoutput><p><B><I>#locale#</I></B><br>
  Locale-specific formats:
  <br>#LSDateFormat(Now(), "mmm-dd-yyyy")# #LSTimeFormat(Now())#<br>
  #LSDateFormat(Now(), "mm/dd/yyyy")# #LSTimeFormat(Now())#<br>
  #LSDateFormat(Now(), "mmm d, yyyy")# #LSTimeFormat(Now())#<br>
  #LSDateFormat(Now(), "d-mmm-yyyy")# #LSTimeFormat(Now())#<br>
  #LSDateFormat(Now(), "dd, mmmm dd, yyyy")# #LSTimeFormat(Now())#<br>
  Standard Date/Time:<br>
  #LSParseDateTime("#LSDateFormat(Now())# #LSTimeFormat(Now())#")#<br>
  </cfoutput>
</cfloop>
LSParseEuroCurrency

Description

Formats a locale-specific currency string as a number. Attempts conversion through each of the default currency formats (none, local, international). Ensures correct handling of euro currency for Euro zone countries.

Returns

A formatted number that matches the value of the string.

Category

International functions, String functions

Function syntax

```
LSParseEuroCurrency(currency-string [, locale])
```

See also

LSParseCurrency, LSEuroCurrencyFormat, SetLocale; Locale-specific content in the Developing ColdFusion Applications

History

ColdFusion 8: Added the locale parameter. ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java locale formatting rules on all platforms, except that it uses the rule detailed in the Usage section for countries in the Euro currency zone.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>currency-string</td>
<td>Locale-specific string or a variable that contains one.</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>

Usage

This function determines whether the current locale's country belongs to the Euro Zone, whose members have converted to the euro; if so, the currency-string parameter must be formatted in euros on all JVMs, including Sun JVM 1.3.1. If the country is not in the Euro zone, the string must follow the locale formatting rules of the JVM. For examples of valid currency formats in all supported locales, see LSEuroCurrencyFormat. For a list of the locale options that ColdFusion supports, and information on setting the default display format of date, time, number, and currency values, see SetLocale.

Example
<h3>LSParseEuroCurrency Example</h3>
<p>Loop through all available locales. Create string representations of the value 123,456 in the three supported currency formats, and parse the results back to numbers.</p>
<cfloop list="#Server.Coldfusion.SupportedLocales#" index="locale" delimiters="",">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput>
        Current Locale: <b><i>#locale#</i></b><br>
        Value in local currency: #localCurrency#<br>
        Parsed using LSParseEuroCurrency:
        #LSParseEuroCurrency(localCurrency)#<br>
    </cfoutput>
    <cfset IntlCurrency = LSEuroCurrencyFormat(123456, "international")>
    Value with International currency formatting: #IntlCurrency#<br>
    Parsed using LSParseEuroCurrency:
    #LSParseEuroCurrency(IntlCurrency)#<br>
    <cfset Currency = LSEuroCurrencyFormat(123456, "none")>
    Value with no currency formatting: #currency#<br>
    Parsed using LSParseEuroCurrency:
    #LSParseEuroCurrency(Currency)#<br>
</cfoutput>
</cfloop>
LSParseNumber

Description

Converts a string that is a valid numeric representation in the current locale into a formatted number.

Returns

A formatted number that matches the value of the string.

Category

International functions, String functions

Function syntax

LSParseNumber(string [, locale])

See also

LSParseDateTime, SetLocale; Locales in the Developing ColdFusion Applications

History

ColdFusion 8: Added the locale parameter. ColdFusion MX: Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>

Usage

This function uses Java standard locale formatting rules on all platforms. To set the default display format of date, time, number, and currency values, use the SetLocale function.

Example
<h3>LSParseNumber Example</h3>
<p>LSParseNumber converts a locale-specific string to a number. Returns the number matching the value of string.</p>
<!---- loop through a list of locales and show number values ---->
<cfloop LIST = "#{Server.Coldfusion.SupportedLocales#}"
    index = "locale" delimiters = ","=">
    <cfset oldlocale = SetLocale(locale)>
    <cfoutput><p><B><I>#locale#</I></B><br>
#LSNumberFormat(-1234.5678, "_________")#<br>
#LSNumberFormat(-1234.5678, "_________.___")#<br>
#LSNumberFormat(1234.5678, "_________")#<br>
#LSNumberFormat(1234.5678, "_________.___")#<br>
#LSNumberFormat(1234.5678, "$_(_________.___)" )#<br>
#LSNumberFormat(-1234.5678, "$_(_________.___)" )#<br>
#LSNumberFormat(-1234.5678, "+_________.___")#<br>
#LSNumberFormat(-1234.5678, "-_________.___")#<br>
The actual number:<br>
#LSParseNumber(LSNumberFormat(1234.5678, "_________"))#<br>
</cfoutput>
</cfloop>
LSTimeFormat

Description

Formats the time part of a date/time string into a string in a locale-specific format.

Returns

A string representing the time value.

Category

Date and time functions, Display and formatting functions, International functions

Function syntax

LSTimeFormat(time [, mask ])

See also

LSParseDateTime, LSDateFormat, TimeFormat: Locales in the Developing ColdFusion Applications

History

ColdFusion MX 6.1: Added the mask character L or l to represent milliseconds.

ColdFusion MX:

- Changed formatting behavior: this function might return different formatting than in earlier releases. This function uses Java standard locale formatting rules on all platforms.
- Added support for the following mask parameter options: short, medium, long, and full.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| string    | - A date/time value  
|           | - A string that is convertible to a time value  
|           | - A date/time object is in the range 100 AD-9999 AD. |
Masking characters that determine the format:

- h: Hours; no leading zero for single-digit hours (12-hour clock)
- hh: Hours; leading zero for single-digit hours. (12-hour clock)
- H: Hours; no leading zero for single-digit hours (24-hour clock)
- HH: Hours; leading zero for single-digit hours (24-hour clock)
- m: Minutes; no leading zero for single-digit minutes
- mm: Minutes; leading zero for single-digit minutes
- s: Seconds; no leading zero for single-digit seconds
- ss: Seconds; leading zero for single-digit seconds
- l: Milliseconds
- t: One-character time marker string, such as A or P.
- tt: Multiple-character time marker string, such as AM or PM

The following conform to Java locale-specific time encoding standards. Their exact formats depend on the locale:

- short: includes hours, minutes; may include AM or PM
- medium: includes hours, minutes; may include AM or PM
- long: medium plus time zone
- full: long, may also include an hour designator

The default value is short.

Usage

This function uses Java standard locale formatting rules on all platforms. When passing date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object. To calculate a difference between time zones, use the `GetTimeZoneInfo` function. To set the default display format of date, time, number, and currency values, use the `SetLocale` function. If no seconds value is passed to this function, and the mask value is s, the default output seconds format is one zero; for example, `lstimeformat(6:39, "h:m:s")` returns 6:39:0. If the mask value is ss, it returns 6:39:00.

Example
<h3>LSTimeFormat Example</h3>

<p>LSTimeFormat returns a time value using the locale convention.</p>

<!--- loop through a list of locales and show time values --->
<cfloop LIST = "#Server.Coldfusion.SupportedLocales#"
index = "locale" delimiters = ",,">
  <cfset oldlocale = SetLocale(locale)>
  <cfoutput>
    <p><b><i>#locale#</i></b><br>
    #LSTimeFormat(Now())#<br>
    #LSTimeFormat(Now(), hh:mm:ss)#<br>
    #LSTimeFormat(Now(), hh:mm:sst)#<br>
    #LSTimeFormat(Now(), hh:mm:sstt)#<br>
    #LSTimeFormat(Now(), HH:mm:ss)#<br>
    <hr noshade>
  </cfoutput>
</cfloop>

</cfloop>
**LTrim**

**Description**

Removes leading spaces from a string.

**Returns**

A copy of the string, without leading spaces.

**Category**

Display and formatting functions, String functions

**Function syntax**

```
LTrim(string)
```

**See also**

RTrim, ToBase64, Trim

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| string    | A string or a variable that contains one

**Example**

```
<h3>LTrim Example</h3>

<cfif IsDefined("FORM.myText")>
<cfoutput>
<pre>
Your string: "#FORM.myText#"
Your string: "#LTrim(FORM.myText)#"
(left trimmed)
</pre>
</cfoutput>
</cfif>

<form action = "ltrim.cfm">
<p>Type in some text, and it will be modified by LTrim to remove leading spaces from the left</p>
<p><input type = "Text" name = "myText" value = "TEST"></p>
<p><input type = "Submit" name = ""></p>
</form>
```
**ListEach**

**Description**

Iterates over every element of a List object and can call a UDF function, passed as the second argument.

**Returns**

None

**Category**

List functions

**Function syntax**

```
ListEach(String str, UDFMethod function [, String delim, boolean includeEmptyFields])
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>An input list object.</td>
</tr>
<tr>
<td>function</td>
<td>UDF or closure object.</td>
</tr>
<tr>
<td>delim</td>
<td>A list delimiter to be used. The default value is comma (,).</td>
</tr>
<tr>
<td>includeEmptyFields</td>
<td>Boolean. Whether to allow empty fields. Default is false.</td>
</tr>
</tbody>
</table>

**Example 1**

```
<cfscript>

    empArray = ["awdhesh", "kumar", "agrawal"];
    listS = "'awdhesh', 'kumar', 'agrawal'";
    
    ArrayEach(empArray, xclosure);
    ListEach(listS, xclosure);
    
    function xclosure(empname, index)
    {
        writeOutput(empName & " at index: " & index);
    }

</cfscript>
```
Also, the original array can also be passed to the closure function. So the following code is also valid:

```coldfusion
function xclosure(empname, index, empArray)
{
}
...
```

Example 2

```coldfusion
<cfscript>
    cityList = "San Jose, New York, Boston, Las Vegas";

    function printCity(String city)
    {
        writeOutput("<br>Current city: 
    ");
    }

    ListEach(cityList, printCity);
</cfscript>
```
ListReduce

Description

Iterates over each item of the list and calls the closure to work on the item. This function will reduce the list to a single value and will return the value.

Returns

any

Syntax

ListReduce(list, function(result, item [,index, list]) [,initialValue, delimiter, includeEmptyFields])

History

ColdFusion 11: Added this function.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Required</td>
<td></td>
<td>The input list.</td>
</tr>
<tr>
<td>function</td>
<td>Required</td>
<td></td>
<td>Closure or a function reference that will be called for each of the iteration. The arguments passed to the callback are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• item: value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• index : current index for the iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• list : reference of the original list</td>
</tr>
<tr>
<td>initialValue</td>
<td>Optional</td>
<td></td>
<td>Initial value which will be used for the reduce operation. The type is any.</td>
</tr>
<tr>
<td>delimiter</td>
<td>Optional</td>
<td>comma (,)</td>
<td>The list delimiter The type is string.</td>
</tr>
<tr>
<td>includeEmptyFields</td>
<td>Optional</td>
<td>false</td>
<td>Include empty values. The type is boolean.</td>
</tr>
</tbody>
</table>

Example
<cfscript>

list = "red, blue, green, yellow";
ucaseList = listMap(list, function(item)
{
    return ucase(item);
});

writeDump(ucaseList);
writeOutput("<br>");

concatValues = listReduce(list, function(result, item, index, mylist, delimiter)
{
    result = result?:"";
    if(index == 1)
    {
        delimiter = "";
        result&= delimiter & item;
    }
    return result;
});
writeDump(concatValues);

</cfscript>
ListMap

Description

Iterates over every entry of the List and calls the closure function to work on the item of the list. The returned value will be set at the same index in a new list and the new list will be returned.

Returns

List

Syntax

```
ListMap(list, function(item [,index, list]) [,delimiter, includeEmptyFields)
```

History

ColdFusion 11: Added this function.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Required</td>
<td></td>
<td>The input list.</td>
</tr>
</tbody>
</table>
| function           | Required|            | Closure or a function reference that will be called for each of the iteration. The arguments passed to the callback are:  
|                    |         |            | • item: value  
|                    |         |            | • index : current index for the iteration  
|                    |         |            | • list : reference of the original list                                      |
| initialValue       | Optional|            | Initial value which will be used for the reduce operation. The type is any. |
| delimiter          | Optional| comma (,)  | The list delimiter The type is string..                                    |
| includeEmptyFields | Optional| false      | Include empty values. The type is boolean.                                 |
Functions m-r

- Max
- Mid
- Min
- Minute
- Month
- MonthAsString
- Now
- NumberFormat
- ObjectEquals
- ObjectLoad
- ObjectSave
- OnWSAuthenticate
- ORMClearSession
- ORMCloseAllSessions
- ORMCloseSession
- ORMEvictCollection
- ORMEvictEntity
- ORMEvictQueries
- ORMExecuteQuery
- ORMFlush
- ORMFlushAll
- ORMGetSession
- ORMGetSessionFactory
- ORMIndex
- ORMIndexPurge
- ORMReload
- ORMSearch
- ORMSearchOffline
- ParagraphFormat
- ParameterExists
- ParseDateTime
- Pi
- PrecisionEvaluate
- Quarter
- PreserveSingleQuotes
- QueryAddColumn
- QueryAddRow
- QueryNew
- QueryConvertForGrid
- QuerySetCell
- QuotedValueList
- Rand
- Randomize
- RandRange
- ReEscape
- REFind
- REFIndNoCase
- REMatch
- REMatchNoCase
- ReleaseComObject
- RemoveCachedQuery
• RemoveChars
• RepeatString
• Replace
• ReplaceList
• ReplaceNoCase
• REPLace
• REPLaceNoCase
• RestDeleteApplication
• RestResponse
• RestInitApplication
• Reverse
• Right
• RJustify
• Round
• RTrim
• QueryGetRow
• QueryExecute
**Max**

**Description**

Determines the greater of two numbers.

**Returns**

The greater of two numbers.

**Category**

[Mathematical functions]

**Function syntax**

```
Max(number1, number2)
```

**See also**

[Min]

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number1, number2</td>
<td>Numbers</td>
</tr>
</tbody>
</table>

**Example**

```
<!--- basic usage --->
<cfoutput>#max(17,19)#</cfoutput> <!--- outputs 19 --->

<!--- example usage --->
<!--- code extract from a game application --->
<cfset highScore = max(highScore, currentGameScore)> <!--- whichever is higher becomes the new highScore --->
```
**Mid**

**Description**

Extracts a substring from a string.

**Returns**

A string; the set of characters from string, beginning at start, of length count.

**Category**

String functions

**Function syntax**

```
Mid(string, start, count)
```

**See also**

Left, Len, Right

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one. Must be single- quotation mark or double- quotation mark delimited.</td>
</tr>
<tr>
<td>start</td>
<td>A positive integer or a variable that contains one. Position at which to start count. Positions start with 1, not 0.</td>
</tr>
<tr>
<td>count</td>
<td>A positive integer or a variable that contains one. Number of characters to return. (Zero is not valid, but it does not throw an error.)</td>
</tr>
</tbody>
</table>

**Example**

```
<h3>Mid Example</h3>

<cfif IsDefined("Form.myText")>
  <!--- If len returns 0 (zero), then show error message. --->
  <cfif Len(Form.myText)>
    <cfif Len(Form.myText) LTE Form.RemoveChars>
      <cfoutput><p style="color: red; font-weight: bold">Your string #Form.myText# only has #Len(Form.myText)# characters. You cannot output the #Form.removeChars# middle characters of this string because it is not long enough.</p></cfoutput>
    <cfelseif Form.startPos GTE Len(Form.myText)>
      <cfoutput><p style="color: red; font-weight: bold">Your string #Form.myText# only has #Len(Form.myText)# characters. You cannot start at position #Form.startPos#.</p></cfoutput>
    </cfelse>
    <cfoutput>Your original string: <strong>#Form.myText#</strong></cfoutput>
    <cfoutput>Your changed string, showing only the <strong>#Form.removeChars#</strong> middle characters: <strong>#Mid(Form.myText, Form.startPos, Form.removeChars)#</strong></cfoutput>
  </cfif>
  <cfelse>
    <p style="color: red; font-weight: bold">Please enter a string of more than 0 (zero) characters.</p>
  </cfif>
</cfif>

<form action="#CGI.ScriptName#" method="POST">
  <p>Type in some text<br />
  <input type="Text" name="myText"></p>
  <p>Enter a starting position (from the beginning of the entered text)<br />
  <input name="startPos" type="text" size="1"></p>
  <p>How many characters do you want to show?
  <select name="RemoveChars">
    <option value="1">1</option>
    <option value="3" selected>3</option>
    <option value="5">5</option>
    <option value="7">7</option>
    <option value="9">9</option>
  </select>
  <input type="Submit" name="Submit" value="Remove characters"></p>
</form>
Min

Description
Determines the lesser of two numbers.

Returns
The lesser of two numbers.

Category
Mathematical functions

Function syntax
Min(number1, number2)

See also
Max

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number1, number2</td>
<td>Numbers</td>
</tr>
</tbody>
</table>

Example

```html
<h3>Min Example</h3>
<cfif IsDefined("FORM.myNum1")>
  <cfif IsNumeric(FORM.myNum1) and IsNumeric(FORM.myNum2)>
    <p>The maximum of the two numbers is #Max(FORM.myNum1, FORM.myNum2)#</p>
    <p>The minimum of the two numbers is #Min(FORM.myNum1, FORM.myNum2)#</p>
  <cfelse>
    <p>Please enter two numbers</p>
  </cfif>
</cfif>

<form action = "min.cfm">
  <h3>Enter two numbers, and see the maximum and minimum of the two numbers</h3>
  Number 1 <input type = "Text" name = "MyNum1">
  <br>Number 2 <input type = "Text" name = "MyNum2">
  <br><input type = "Submit" name = "" value = "See results">
</form>
```
**Minute**

**Description**

Extracts the minute value from a date/time object.

**Returns**

The ordinal value of the minute, in the range 0-59.

**Category**

Date and time functions

**Function syntax**

```plaintext
Minute(date)
```

**See also**

DatePart, Hash, Second

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

**Usage**

When passing a date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

**Example**

```cfoutput
<h3>Minute Example</h3>

The time is currently #TimeFormat(Now())#.  
We are in hour #Hour(Now())#, Minute #Minute(Now())#  
and Second #Second(Now())# of the day.
</cfoutput>
```
Month

Description

Extracts the month value from a date/time object.

Returns

The ordinal value of the month, in the range 1 (January) - 12 (December).

Category

Date and time functions

Function syntax

```
Month(date)
```

See also

DatePart, MonthAsString, Quarter

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Date/time object, in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

When passing a date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

⚠️ Note

You can pass the CreateDate function or the Now function as the date parameter of this function; for example: #Month(CreateDate(2001, 3, 3))#.

Example
<h3>Month Example</h3>

<!--template:cfstart-->
<cfif IsDefined("FORM.year")>

More information about your date:

<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>

<cfoutput>

<p>Your date, #DateFormat(yourDate)#.</p>
<br>
It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day #DayOfWeek(yourDate)# in the week.
<br>
This is day #Day(yourDate)# in the month of #MonthAsString(Month(yourDate))#, which has #DaysInMonth(yourDate)# days.
<br>
We are in week #Week(yourDate)# of #Year(yourDate)# (day #DayOfYear(yourDate)# of #DaysInYear(yourDate)#). <br>
<cfif IsLeapYear(Year(yourDate))>
This is a leap year
<cfelse>
This is not a leap year
</cfif>

</cfoutput>

<!--template:cfend-->
**MonthAsString**

**Description**

Determines the name of the month that corresponds to `month_number`.

**Returns**

A string; the name of the specified month, in the current locale.

**Category**

Date and time functions, String functions

**Function syntax**

```coldfusion
MonthAsString(month_number [, locale])
```

**See also**

DatePart, Month, Quarter

**History**

ColdFusion 8: Added the `locale` parameter.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>month_number</td>
<td>An integer in the range 1 - 12.</td>
</tr>
<tr>
<td>locale</td>
<td>Locale to use instead of the locale of the page when processing the function</td>
</tr>
</tbody>
</table>

**Example**
<h3>MonthAsString Example</h3>

<cfif IsDefined("FORM.year")>
<p>More information about your date:</p>
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>

<cfoutput>
<p>Your date, #DateFormat(yourDate)#.  
It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day #DayOfWeek(yourDate)# in the week.  
This is day #Day(YourDate)# in the month of #MonthAsString(Month(yourDate))#, which has #DaysInMonth(yourDate)# days.  
We are in week #Week(yourDate)# of #Year(yourDate)# (day #DayOfYear(yourDate)# of #DaysInYear(yourDate)#).  
<cfif IsLeapYear(Year(yourDate))>This is a leap year  
<cfelse>This is not a leap year  
</cfif>
</cfoutput>
</cfif>
**Now**

**Description**

Gets the current date and time of the computer running the ColdFusion server. The return value can be passed as a parameter to date functions such as `DaysInYear` or `FirstDayOfMonth`.

**Returns**

A date/time object; the current date and time of the computer running the ColdFusion server.

**Category**

*Date and time functions*

**Function syntax**

```
Now()
```

**See also**

`CreateDateTime`, `DatePart`

**Example**

```
<h3>Now Example</h3>
<p>Now returns the current date and time as a valid date/time object.</p>

<p>The current date/time value is <cfoutput>#Now()#</cfoutput></p>
<p>You can also represent this as <cfoutput>#DateFormat(Now())#</cfoutput>, <cfoutput>#TimeFormat(Now())#</cfoutput></p>
```
NumberFormat

Description

Creates a custom-formatted number value. Supports the numeric formatting used in the U.S. For international number formatting, see LSNumberFormat.

Returns

A formatted number value:

- If no mask is specified, returns the value as an integer with a thousands separator.
- If the parameter value is "" (an empty string), returns 0.

Category

Display and formatting functions

Function syntax

```
NumberFormat(number [, mask ])
```

See also

DecimalFormat, DollarFormat, IsNumeric, LSNumberFormat

History

ColdFusion MX: Changed behavior: if the mask format cannot correctly mask a number, this function returns the number unchanged. (It does not truncate the number nor throw an error.) (If no mask is selected, ColdFusion MX rounds the decimal part as ColdFusion 5 does. For example, it rounds 34.567 to 35.)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A number.</td>
</tr>
<tr>
<td>mask</td>
<td>A string or a variable that contains one. Set of characters that determine how ColdFusion displays the number</td>
</tr>
</tbody>
</table>

The following table explains mask characters:

<table>
<thead>
<tr>
<th>Mask character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ (underscore)</td>
<td>Optional. Digit placeholder.</td>
</tr>
<tr>
<td>9</td>
<td>Optional. Digit placeholder. (Shows decimal places more clearly than <em>). Note that when ‘9’ or ‘</em>’ is used after the decimal place, 0 will be padded, if required.</td>
</tr>
<tr>
<td>Mask</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>.</td>
<td>Location of a mandatory decimal point.</td>
</tr>
<tr>
<td>0</td>
<td>Located to the left or right of a mandatory decimal point. Pads with zeros.</td>
</tr>
<tr>
<td>( )</td>
<td>If number is less than zero, puts parentheses around the mask.</td>
</tr>
<tr>
<td>+</td>
<td>Puts plus sign before positive number; minus sign before negative number.</td>
</tr>
<tr>
<td>-</td>
<td>Puts a space before positive number; minus sign before negative number.</td>
</tr>
<tr>
<td>,</td>
<td>Separates every third decimal place with a comma.</td>
</tr>
<tr>
<td>L,C</td>
<td>Left-justifies or center-justifies number within width of mask column. First character of mask must be L or C. The default value is right-justified.</td>
</tr>
<tr>
<td>$</td>
<td>Puts a dollar sign before formatted number. First character of mask must be the dollar sign ($).</td>
</tr>
<tr>
<td>^</td>
<td>Separates left and right formatting.</td>
</tr>
</tbody>
</table>

**Note**

If you do not specify a sign for the mask, positive and negative numbers do not align in columns. To put a plus sign or space before positive numbers and a minus sign before negative numbers, use the plus or minus sign, respectively.

**Usage**

This function uses Java standard locale formatting rules on all platforms. The position of symbols in format masks determines where the codes take effect. For example, if you put a dollar sign at the far left of a format mask, ColdFusion displays a dollar sign at the left edge of the formatted number. If you separate the dollar sign on the left edge of the format mask by at least one underscore, ColdFusion displays the dollar sign just to the left of the digits in the formatted number. These examples show how symbols determine formats:

<table>
<thead>
<tr>
<th>Number</th>
<th>Mask</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.37</td>
<td>$__<em>.</em></td>
<td>&quot;$ 4.37&quot;</td>
</tr>
<tr>
<td>4.37</td>
<td>$.--</td>
<td>&quot;$ 4.37&quot;</td>
</tr>
</tbody>
</table>

The positioning can also show where to place the minus sign for negative numbers:
The positions for a symbol are: far left, near left, near right, and far right. The left and right positions are determined by the side of the decimal point on which the code character is shown. For formats that do not have a fixed number of decimal places, you can use a caret (^) to separate the left fields from the right. An underscore determines whether the code is placed in the far or near position. Most code characters’ effect is determined by the field in which they are located. This example shows how to specify where to put parentheses to display negative numbers:

<table>
<thead>
<tr>
<th>Number</th>
<th>Mask</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.21</td>
<td>C(<em>^</em>)</td>
<td>&quot;(3.21)&quot;</td>
</tr>
<tr>
<td>3.21</td>
<td>C_(^_)</td>
<td>&quot;(3.21)&quot;</td>
</tr>
<tr>
<td>3.21</td>
<td>C(_^)</td>
<td>&quot;(3.21)&quot;</td>
</tr>
<tr>
<td>3.21</td>
<td>C_(^)</td>
<td>&quot;(3.21)&quot;</td>
</tr>
</tbody>
</table>

When converting from string to double, to prevent rounding errors, this function adds a rounding factor of 1.5543122344752E-014 to the converted number. For example, without adding the rounding factor, converting the string value 1.275 to double with two digits of precision results in a value of 1.27499999999999999, which would be rounded up to 1.27. By adding the rounding factor, the conversion correctly results in a value of 1.28. If you round off a double such as 1.99499999999999999999999999999, where the last decimal is 10E-14, the rounding factor can cause an incorrect result. To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

Example

```xml
<h3>NumberFormat Example</h3>

<cfloop FROM = 1000 TO = 1020 INDEX = "counter">
<cfset CounterRoot2 = counter * sqr(2)>

<!--- Show result in default format, adding comma for thousands place; and in custom format, displaying to two decimal places --->
<cfoutput>
<pre>#counter# * Square Root of 2: #NumberFormat(CounterRoot2, _____.__)#</pre>
<pre>#counter# * Square Root of 2: #NumberFormat(CounterRoot2)#</pre>
</cfoutput>
</cfloop>"
ObjectEquals

Description

Identifies any conflict between CFC instances on the client and the server.

Returns

Category

Other functions

Function Syntax

```
objectEquals( Param1 ,Param2)
```

See Also

Conflict management in Developing ColdFusion Applications.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>param 1</td>
<td>New instance of the CFC on the client.</td>
</tr>
<tr>
<td>param 2</td>
<td>Original instance of the CFC.</td>
</tr>
</tbody>
</table>

Example

```
<cfset isNotConflict = ObjectEquals(originalobject, serverobject)>
<cfif isNotConflict>
<cfif operation eq "UPDATE">
<cfset obj = ORMGetSession().merge(clientobject)>
<cfset EntitySave(obj)>
<cfelseif operation eq "DELETE">
<cfset obj = ORMGetSession().merge(originalobject)>
<cfset EntityDelete(obj)>
</cfif>
<cfelse><!-----Conflict----->
<cflog text = "is a conflict">
<cfset conflict = CreateObject("component","CFIDE.AIR.conflict")>
<cfset conflict.serverobject = serverobject>
<cfset conflict.clientobject = clientobject>
<cfset conflict.originalobject = originalobject>
<cfset conflict.operation = operation>
<cfset conflicts[conflictcount++] = conflict>
<cfcontinue>
```
ObjectLoad

Description

Loads a serialized ColdFusion array, CFC, DateTime object, Java object, query, or structure into memory as the corresponding object.

Returns

The deserialized ColdFusion object, such as a CFC or a query object.

Category

Other functions

Function syntax

```
ObjectLoad(binaryObject)
ObjectLoad(filepath)
```

See also

ObjectSave

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binaryObject</td>
<td>A binary object returned by ObjectSave function.</td>
</tr>
<tr>
<td>filepath</td>
<td>A string specifying the path to a file containing a serialized complex object, such as a query or CFC, or a variable that is a serializable binary representation of a complex object. This parameter must be the name of a file or an object returned by the SaveCFOBJECT function.</td>
</tr>
</tbody>
</table>

Usage

This function is useful for handling dynamic data that has a relatively long period of usefulness and takes substantial time or resources to obtain. It lets you save the data in a file and use it in multiple application instances. For example, you can create a CFC that stores a query that takes long time to run and retrieves infrequently updated data. If you use the ObjectSave function to initially save the CFC as a file, you can then deserialize the CFC file on future application starts and improve application performance.

Example
<h3>Loading and saving an object.</h3>

<!--- Create the component object. --->
<cfobject component="tellTime" name="tellTimeObj">
<!--- Save the component object to a file. --->
<cfset ObjectSave(tellTimeObj, "data.out")/>
<!--- Load the component object again. --->
<cfset ObjLoaded = ObjectLoad("data.out")>
<!--- Invoke the methods from loaded objects. --->
<cfinvoke component="#ObjLoaded#" method="getLocalTime" returnvariable="localTime">
<cfinvoke component="#ObjLoaded#" method="getUTCTime" returnvariable="UTCTime">
<!--- Display the results. --->
<h3>Time Display Page</h3>
<cfoutput>
Server's Local Time: #localTime#
Calculated UTC Time: #UTCTime#
</cfoutput>
ObjectSave

Description

Converts a ColdFusion array, CFC, DateTime object, Java object, query, or structure into a serializable binary object and optionally saves the object in a file.

Returns

A serializable binary representation of the object.

Category

Other functions

Function syntax

ObjectSave(object[, filePath])

See also

ObjectLoad

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The complex object, such as a query or CFC, that will be serialized.</td>
</tr>
<tr>
<td>filePath</td>
<td>The path of the file in which to save the serialized data.</td>
</tr>
</tbody>
</table>

Usage

This function is useful for handling dynamic data that has a relatively long period of usefulness and takes substantial time or resources to obtain. It lets you save the data in a file and use it in multiple application instances. For example, you can create a CFC that stores a query that takes long time to run and retrieves infrequently updated data. If you use the ObjectSave function to initially save the CFC as a file, and deserialize the CFC file on future application starts, you can improve application performance.

Example
<h3>Saving and loading an object</h3>

<!--- Create the component object. --->
<cfobject component="tellTime" name="tellTimeObj">

<!--- Save the component object to a file. --->
<cfset ObjectSave(tellTimeObj, "data.out")/>

<!--- Load the component object again. --->
<cfset ObjLoaded = ObjectLoad("data.out")>

<!--- Invoke the methods from loaded objects. --->
<cfinvoke component="#ObjLoaded#" method="getLocalTime" returnvariable="localTime">
<cfinvoke component="#ObjLoaded#" method="getUTCTime" returnvariable="UTCTime">

<!--- Display the results. --->
<h3>Time Display Page</h3>
<cfoutput>
Server's Local Time: #localTime#
Calculated UTC Time: #UTCTime#
</cfoutput>
OnWSAuthenticate

Description

Authenticates the user

History

ColdFusion 10: Added this function

Syntax

OnWSAuthenticate(username, password, connectionInfo)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Name of the user that has to be authenticated.</td>
</tr>
<tr>
<td>password</td>
<td>Password for the user.</td>
</tr>
<tr>
<td>connectionInfo</td>
<td>A struct that contains the following keys:</td>
</tr>
<tr>
<td></td>
<td>• Authenticated: {{YES, NO}}</td>
</tr>
<tr>
<td></td>
<td>• ConnectionTime: Connection time stamp.</td>
</tr>
<tr>
<td></td>
<td>• clientID: Unique ID of the client. Custom keys are also supported.</td>
</tr>
</tbody>
</table>
|                  | For example, you can specify the user's role, status, or age. The connectionInfo is shared across all the channels for a given WebSocket client. Also, modifications are persisted across all the subscriptions for that client.

Example

The following example uses the function onWSAuthenticate, validates the user, and associates a user role.

⚠️ Note

For this example to work, ensure that you implement the user-defined functions.
component
{
    this.name="websocketsampleapp23";
    this.wschannels=[{name="stocks",cfclistener="stocksListener"}];

    function onWSAuthenticate(username, password, connectionInfo)
    {
        //write appropriate logic to fetch user password in function checkPassword
        If(checkPassword(username) eq password)
        {
            connectionInfo.authenticated="YES";
            //Role is the custom information that you provide
            connectionInfo.role= "admin";
            return true;
        }
        else{
            connectionInfo.authenticated="NO";
            return false;
        }
        writedump("#connectionInfo#","console");
    }
}
ORMClearSession

ORMClearSession removes all the entities that are loaded or created in the session. This clears the first level cache and removes the objects that are not yet saved to the database.
ORMCloseAllSessions

Description
Closes all Hibernate sessions in the request.

Function Syntax
ormcloseallsessions()

See Also
ORMGetSession, ORMClearSession, ORMFlush, ORMGetSessionFactory, ORMCloseSession in Developing ColdFusion Applications.
ORMCloseSession

Description

Closes the Hibernate session associated with the data source in the request. If you do not specify a data source, the Hibernate session associated with the default data source is closed.

Category

ORM functions

Function Syntax

ormclosesession(datasource)

See Also

ORMGetSession, ORMClearSession, ORMFlush, ORMGetSessionFactory, ORMCloseSession in Developing ColdFusion Applications.
**ORMEvictCollection**

**Description**

This method is used to evict all the collection or association data for the given entity name and collection name, from the secondary cache. If the primary key is specified, then, the collection or association data of the entity with the primary key is evicted.

**Category**

**ORM functions**

**See Also**

ORMEvictEntity, ORMEvictQueries, Evict content from secondary cache in Caching in the Developing ColdFusion Applications

**Function Syntax**

```
ormevictcollection("<entity_name>", "<collection_name>", [primarykey])
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity name</td>
<td>Entity name of the persistent CFC.</td>
</tr>
<tr>
<td>collection name</td>
<td>Name of the collection in the component</td>
</tr>
<tr>
<td>primary key</td>
<td>Primary key of the collection or association data of the entity</td>
</tr>
</tbody>
</table>

**Example**

To evict all the association or collection data of collection arts belonging to the component CArtists:

```
<cfset ORMEvictCollection("CArtists", "arts")>
```
**ORMEvictEntity**

**Description**

This method is used to evict items for the given entity name, from the secondary cache. If the primary key is specified, then the data of the entity with that primary key is evicted. Primary key should be a value in case of simple primary key or should be a struct in case of composite primary key.

**Category**

ORM functions

**See Also**

ORMEvictCollection, ORMEvictQueries, Evict content from secondary cache in Caching in the Developing ColdFusion Applications

**Function Syntax**

```coldfusion
ORMEvictEntity("<entity_name>", [primarykey])
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component name</td>
<td>Entity name of the persistent CFC</td>
</tr>
<tr>
<td>primary key</td>
<td>Primary key value of the component</td>
</tr>
</tbody>
</table>

**Example**

To evict all the cache data of CArtist entity:

```coldfusion
<cfset ORMEvictEntity("CArtists")>
```

To evict the cache data of CArtists entity whose primary key is 1:

```coldfusion
<cfset ORMEvictEntity("CArtists", 1)>
```
**ORMEvictQueries**

**Description**

This method is used to evict the data of all the queries from the default query cache of the specified data source. If cache name is specified, then the data of all queries belonging to the cache region with the given cache name are evicted. If no data source is specified, the default query cache of the default data source is evicted.

**Category**

ORM functions

**See Also**

ORMEvictEntity, ORMEvictCollection, Evict content from secondary cache in Caching in the Developing ColdFusion Applications

**Syntax**

ORMEvictQueries(cachename) ORMEvictQueries(cachename, datasource)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachename</td>
<td>Name of the cache region that you want to evict.</td>
</tr>
<tr>
<td>datasource</td>
<td>Name of the data source whose cache you want to evict. If you do not specify the cache, the default query cache is evicted.</td>
</tr>
</tbody>
</table>

**Example**

Evicts the data of all the queries from the default query cache.

```cfset ORMEvictQueries()>
```

Evicts the data of all the queries from the cache region with the name availableArtsCache.

```cfset ORMEvictQueries("availableArtsCache")>
```
ORMExecuteQuery

Description

Executes a Hibernate Query Language (HQL) query. By default, this function works on ORM's default data source. To use this function for another data source, specify the data source key-value pair within the queryoptions.

Syntax

ORMExecuteQuery(hql, params [,unique])

ORMExecuteQuery(hql, [,unique] [, queryoptions])

ORMExecuteQuery(hql, params [,unique] [,queryOptions])

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hql</td>
<td>The HQL query that has to be executed.</td>
</tr>
<tr>
<td>Params</td>
<td>Object parameter for the entity.</td>
</tr>
<tr>
<td>Unique</td>
<td>Specifies if the object parameter is unique.</td>
</tr>
<tr>
<td>Queryoptions</td>
<td>Key-value pair of options for the query.</td>
</tr>
</tbody>
</table>

You can specify several options to control the behavior of retrieval using queryOptions:

- `ignorecase`: Ignores the case of sort order when you set it to true. Use this option only when you specify the sortorder parameter.
- `maxResults`: Specifies the maximum number of objects to be retrieved.
- `offset`: Specifies the start index of the resultset from where it has to start the retrieval.
- `cacheable`: Whether the result of this query is to be cached in the secondary cache. Default is false.
- `cachename`: Name of the cache in secondary cache.
- `timeout`: Specifies the timeout value (in seconds) for the query.

Note: Maxresults and timeout are used for pagination.

Example
ORMExecuteQuery("from cemployees where EmployeeID=2");
ORMExecuteQuery("from cemployees where EmployeeID IN (?, ?, ?)", [1, 3, 5]);
ORMExecuteQuery("select count(*) from cemployees", [], false);
ORMExecuteQuery("from Artists where artistid=1", true, {datasource="cfartgallery"});
ORMExecuteQuery("select count(*) from Authors", [], false, {datasource="cfbookclub"});
ORMFlush

Flushes the current ORM session associated with the data source specified in the application. ORMFlush flushes all the pending CRUD operations in that request.
ORMFlushall

Description

Flushes all the current Hibernate sessions in the request.

Function syntax

ormflushall()

See Also

ORMCloseSession, ORMClearSession, ORMGetSession, ORMGetSessionFactory, ORMFlush in Developing ColdFusion Applications.
ORMGetSession

Returns the current ORM session. That is, the Hibernate Session associated with the data source specified in the application. This returns the underlying Hibernate Session that can be used to call the API, which otherwise is not exposed by ColdFusion.
ORMGetSessionFactory

ORMGetSessionFactory returns the underlying Hibernate SessionFactory object.
### ORMIndex

**Description**

Performs offline indexing.

**History**

ColdFusion 10: Added this function

**Syntax**

```
ORMIndex(__); ORMIndex("_entity_name"); ORMIndex("entityName_list"); ORMIndex (entityObject__);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityType</td>
<td>Name of the entity that has to be indexed.</td>
</tr>
<tr>
<td>entityName_list</td>
<td>Comma-separated list of entity names for indexing.</td>
</tr>
<tr>
<td>entityObject</td>
<td>Variable name of a specific entity instance for indexing.</td>
</tr>
</tbody>
</table>

**Usage**

If you use this function without specifying any parameters, all persistent entities of a given application are indexed.

**Example**

```groovy
EmpObjs = EntityLoad("Employee",{lastname="Bond");
for(EmpObj in EmpObjs)
{
    ormindex(EmpObj);
}
```
ORMIndexPurge

Description

Clears all indexed data for all entities or specified entities in the current application scope.

History

ColdFusion 10: Added this function

Syntax

ORMIndexPurge(); ORMIndexPurge("entityName"); ORMIndexPurge("entityName_list");

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>Name of the entity to be loaded.</td>
</tr>
<tr>
<td>entityName_list</td>
<td>Comma-separated list of entity names for purging.</td>
</tr>
</tbody>
</table>

Usage

If you use this function without specifying entityName, all persistent entities of the application are purged.

Example

ORMIndexPurge();

ORMIndexPurge("Employee");
ORMReload

Description

Reinitializes ORM for the application. If you make any change to the persistent metadata of the CFCs, then you might want to reload the ORM.

Returns

Returns ORM session factory instance.

Category

ORM functions

Function Syntax

ORMReload()

Usage

Adobe recommends that you use this function only during the time of development.

See Also

ColdFusion ORM

Example

component
{
    this.name = Hash( GetCurrentTemplatePath() );
    /* define the application wide datasource */
    this.datasource = "cfartgallery";
    /* enable hibernate support for this application */
    this.ormenabled = true;
    /* create a struct of ORM settings */
    this.ormsettings = {};
    /* turn on event handling */
    this.ormsettings.eventhandling = true;
    /**
    * @output true
    */
    public boolean function onRequestStart( targetPage )
    {
        /* this is to ensure that ORM is up-to-date for demo */
        ORMReload();
        return true;
    }
}
ORMSearch

Description

Searches for given text in specific properties or entities.

Returns

A struct that contains the following:

- An array of structs (with the `entity` and `score` being the keys) in the following format:

  ```
  data -[[entity: entity1, score: entity1_score], {entity: entity2, score: entity2_score}, ..... ]
  ```

- `maxTotalRecord` (number of possible results)

History

ColdFusion 10: Added this function

Syntax

ORMSearch("query_text", "entityName") ORMSearch("query_text", "entityName", fields) ORMSearch("query_text", "entityName", fields, optionMap)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>query_text</code></td>
<td>The text to be searched for or a complete Lucene query.In the case of ORMSearch(&quot;query_text&quot;, &quot;entityName&quot;), only Lucene query is supported. For details of Lucene query, see <a href="http://lucene.apache.org/core/old_versioned_docs/versions/3_0_0/queryparsersyntax.html">http://lucene.apache.org/core/old_versioned_docs/versions/3_0_0/queryparsersyntax.html</a></td>
</tr>
<tr>
<td><code>entityName</code></td>
<td>Name of the entity to be searched.</td>
</tr>
<tr>
<td><code>fields</code></td>
<td>Fields in which search has to be performed. This can be an array of strings. If you are performing a Lucene query, you need not specify this field. In other words, if you do not specify this value, a Lucene query is performed. Field name is case-sensitive.</td>
</tr>
</tbody>
</table>
optionMap

Extra options that can be passed while executing Lucene query. The options are:

- **sort**: Sorts based on `indexfieldname` you specified.
- **offSet**: Specifies the position from which to retrieve the objects.
- **maxResults**: Specifies the maximum number of objects to be retrieved.

**Usage**

When you perform a date search, use the format `yyyyymmdd` as shown in the following example:

```coldfusion
objs = ORMSearch("datecheck:[#dateformat(dateadd("d",5,now()),"yyyyymmdd") TO #dateformat(dateadd("d",35,now()),"yyyyymmdd")#","C2",[],{maxresults=2});
```

If you are performing a time search, use the UTC format.

**Example 1: ORM search based on Lucene query**

```coldfusion
ORMSearch("FirstName:ch*","Employee");
ORMSearch("ch*","Employee",["FirstName"]);
objs = ORMSearch('FirstName:ch*','Employee',[],{sort="salary",maxresults=5,offset=2});
```

**Example 2: ORM search on multiple entities**

```coldfusion
ORMSearch("john*","DeveloperEntity,UserEntity",["firstname"]);
```

In this example, first name is searched in the **DeveloperEntity** and **UserEntity** and a composite array of entities are returned.

**Example 3: ORM search on all subentities based on Lucene query**

This example shows how to perform ORM search on all subentities that inherit a super entity. Assume that **USEmployeeEntity** and **UKEmployeeEntity** are extending **EmployeeEntity**. You can search both the subentities using the following code:

```coldfusion
ORMSearch("john*","EmployeeEntity",["FirstName"]);
```

**Example 4: ORM search in relationships**

In this example, products and categories have a many-to-one relationship. You can search all products of a specific category using the following code:
Note that search in related objects works only for many-to-one relationship and one-to-one relationship.
**ORMSearchOffline**

**Description**

Performs search on the indexed properties but returns only the stored fields. For this function to work, specify `index Store=true` on the properties on which you want to perform the search.

**History**

ColdFusion 10: Added this function

**Returns**

A struct that contains the following:

- An array of structs (with the *entity* and *score* being the keys) in the following format:

  ```
  data -[{entity: entity1, score: entity1_score}, {entity: entity2, score: entity2_score}, ..... ]
  ```

- `maxTotalRecord` (number of possible results)
- `fields_to_be_selected` as keys.

**Syntax**

```cpp
ORMSearchOffline(query_text, entityName, fields_to_be_selected);
ORMSearchOffline(query_text, entityName, fields_to_be_selected, fields);
ORMSearchOffline(query_text, entityName, fields_to_be_selected, fields, optionMap);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>query_text</code></td>
<td>The text to be searched for or a complete Lucene query. For details of Lucene query, see <a href="http://lucene.apache.org/core/old_versioned_docs/versions/">http://lucene.apache.org/core/old_versioned_docs/versions/</a>.</td>
</tr>
<tr>
<td><code>entityName</code></td>
<td>Name of the entity to be searched.</td>
</tr>
<tr>
<td><code>fields_to_be_selected</code></td>
<td>Fields to be returned as keys in the resultant struct.</td>
</tr>
<tr>
<td><code>fields</code></td>
<td>Fields in which search has to be performed.</td>
</tr>
<tr>
<td><code>optionMap</code></td>
<td></td>
</tr>
</tbody>
</table>
Extra options that can be passed while executing Lucene query. The options can be:

- **sort**: Sorts based on `indexfieldname` you specified.
- **offset**: Specifies the position from which to retrieve the objects.
- **maxResults**: Specifies the maximum number of objects to be retrieved.

**Example 1**

```coldfusion
ORMSearchOffline('FirstName:”ch*”', "Employee", ["id", "firstname"]);
```

**Example 2**

In the following example, offline search is performed on the property `FirstName` and first name and last name are returned as keys in the resultant struct.

```coldfusion
ORMSearchOffline("ch*", "Employee", ["FirstName", "LastName"], ["FirstName"], {sort="salary", maxresults=5, offset=2});
```

**Example 3**

In this example, the `resultObj` in the query is an array of structs. The individual structs contain all the selected fields (passed as third parameter).

```coldfusion
<cfset resultObj = ORMSearchOffline('Java Rocks', 'Book', [bookId, summary, Author.name, title], [title, short_summary])>
```
ParagraphFormat

Description

Replaces characters in a string:

- Single newline characters (CR/LF sequences) with spaces
- Double newline characters with HTML paragraph tags (&lt;p&gt;)

Returns

A copy of the string, with characters converted.

Category

Display and formatting functions, String functions

Function syntax

```
ParagraphFormat(string)
```

See also

StripCR

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

Usage

This function is useful for displaying data entered in textarea fields.

Example
<h3>ParagraphFormat Example</h3>

<p>Enter text into this textarea, and see it returned as HTML.</p>

<cfif IsDefined("FORM.myTextArea")>
<p>Your text area, formatted</p>
<p><cfoutput>#ParagraphFormat(FORM.myTextArea)#</cfoutput></cfif>

<!--- use #Chr(10)#Chr(13)# to simulate a line feed/carriage return combination; i.e., a return --->
<form action = "paragraphformat.cfm">
<textArea name = "MyTextArea" cols = "35" ROWS = 8>
This is sample text and you see how it scrolls
</textArea>
</form>
<input type = "Submit" name = "Show me the HTML version">
</form>
ParameterExists

Deprecated

THIS FUNCTION HAS BEEN DEPRECATED

Description

This function is deprecated. Use the IsDefined function. Determines whether a parameter exists. ColdFusion does not evaluate the argument.

History

ColdFusion MX: Deprecated this function. It might not work, and might cause an error, in later releases.
ParseDateTime

Description

Parses a date/time string according to the English (U.S.) locale conventions. (To format a date/time string for other locales, use the LSParseDateTime function.)

Returns

A date/time object

Category

Date and time functions, Display and formatting functions

Function syntax

ParseDateTime(date/time-string [, pop-conversion ])

See also

IsDate, IsNumericDate, SetLocale

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date/time string</td>
<td>A string containing a date/time value formatted according to U.S. locale conventions. Can represent a date/time in the range 100 AD-9999 AD. Years 0-29 are interpreted as 2000-2029; years 30-99 are interpreted as 1930-1999.</td>
</tr>
</tbody>
</table>
pop-conversion

- **pop**: specifies that the date/time string is in POP format, which includes the local time of the sender and a time-zone offset from UTC. ColdFusion applies the offset and returns a value with the UTC time.
- **standard**: (the default) function does no conversion.

Apart from these 2 values, this parameter allows you to specify the format in which to parse the given string.

Valid letters that can be used to specify the format are described here [http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html](http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html)

For instance:

```coldfusion
<cfset str = "Fri Jan 03 04:30:10 EDT 2014">
<cfset date = parsedateTime(str,"EEE MMM dd HH:mm:ss zzz yyyy")>
```

Another example:

```coldfusion
<cfset string = "1997-07-16T19:20:30+01:00">
<cfset date = parseDateTime(string, "yyyy-MM-dd'T'HH:mm:ssX")>
```

**Usage**

This function is similar to `CreateDateTime`, but it takes a string instead of enumerated date/time values. These functions are provided primarily to increase the readability of code in compound expressions. To calculate a difference between time zones, use the `GetTimeZoneInfo` function. To set the default display format of date, time, number, and currency values, use the `SetLocale` function.

**Example**
<h3>ParseDateTime Example</h3>
<cfif IsDefined("form.theTestValue")>
  <cfif IsDate(form.theTestValue)>
    <h3>The expression <cfoutput>#DE(form.theTestValue)#</cfoutput> is a valid date</h3>
    <p>The parsed date/time is:<br>
    <cfoutput>#ParseDateTime(form.theTestValue)#</cfoutput></p>
  </cfif>
<cfelse>
  <h3>The expression <cfoutput>#DE(form.theTestValue)#</cfoutput> is not a valid date</h3>
</cfif>
</cfif>
</form>
Pi

Description

Gets the mathematical constant p, accurate to 15 digits.

Returns

The number 3.14159265358979.

Category

Mathematical functions

Function syntax

```
Pi()
```

See also

ASin, Cos, Sin, Tan

Example

```html
<h3>Pi Example</h3>
<!--- By default, ColdFusion displays only 11 significant digits. Use NumberFormat to display all 15. --->
The Pi function Returns the number
<cfoutput>
#NumberFormat(Pi(), "_._______________")#,
</cfoutput> the mathematical constant pi, accurate to 15 digits.
```
PrecisionEvaluate

Description

Evaluates one or more string expressions, dynamically, from left to right, using BigDecimal precision arithmetic to calculate the values of arbitrary precision arithmetic expressions.

Returns

An object; the result of the evaluations.

Category

Mathematical functions, Dynamic evaluation functions

Function syntax

```
PrecisionEvaluate(string_expression1 [, string_expression2 , ... ])
```

See also

Evaluate, Using Expressions and Number Signs in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string_expression1, string_expression2...</td>
<td>Expressions to evaluate</td>
</tr>
</tbody>
</table>

Usage

The PrecisionEvaluate function lets you calculate arbitrarily long decimal (BigDecimal precision) values. BigDecimal precision arithmetic accepts and generates decimal numbers of any length, and does not use exponential notation. The PrecisionEvaluate function calculates arbitrary precision results only for addition, subtraction, multiplication, and division. If you use any of the following operations, ColdFusion performs normal integer or floating point arithmetic and does not return BigDecimal values.

- exponentiation (^)
- modulus (MOD or %)
- integer division

This function differs from the Evaluate function only in its use of BigDecimal precision arithmetic to calculate numeric values; otherwise the two functions are identical. The results of an evaluation on the left can have meaning in an expression to the right, and the function returns the result of evaluating the rightmost expression. If a string expression contains a single- or double-quotation mark, the mark must be escaped. If an expression, such as 1/3, results in an infinitely repeating decimal value, ColdFusion limits the decimal part to 20 digits.
Note

To increase processing efficiency, do not put the arithmetic expressions to evaluate in quotation marks ("."). ColdFusion compiles `PrecisionEvaluate(a*b)` more efficiently than it compiles `PrecisionEvaluate("a*b")`, although both formats produce the same results.

Example

```coldfusion
<h3>PrecisionEvaluate Example</h3>
<cfif IsDefined("FORM.myExpression")>
  <cftry>
  <!--- Evaluate the expression and display the result. --->
  <cfset theExpression = PrecisionEvaluate(Form.myExpression)>
  <cfoutput>
  The value of the expression #FORM.MyExpression# is #theExpression#.<br>
  </cfoutput>

  <cfcatch type="any">
  <cfoutput>Could not evaluate the expression #Form.myExpression#.</cfoutput>
  </cfcatch>
  </cftry>
</cfif>

<cfform preservedata="yes">
  <h3>Enter a ColdFusion expression for evaluation.</h3>
  <p>Try using some really big decimal numbers.</p>
  <cfinput type="text" name="myExpression" size="60"><br>
  <br>
  <cfinput type="submit" name="submit">
</cfform>
```
Quarter

Description

Calculates the quarter of the year in which a date falls.

Returns

An integer, 1-4.

Category

Date and time functions

Function syntax

Quarter(date)

See also

DatePart, Month

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date/time object in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

When passing a date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

Example

<h3>Quarter Example</h3>

Today, <cfoutput>#DateFormat(Now())#</cfoutput>, is in Quarter <cfoutput>#Quarter(Now())#</cfoutput>.
**PreserveSingleQuotes**

**Description**

Prevents ColdFusion from automatically escaping single-quotation mark characters that are contained in a variable. ColdFusion does not evaluate the argument.

**Returns**

(None)

**Category**

Other functions

**Function syntax**

```
PreserveSingleQuotes(variable)
```

**History**

ColdFusion MX: Changed behavior: ColdFusion automatically escapes simple-variable, array-variable, and structure-variable references within a `cfquery` tag body or block. (Earlier releases did not automatically escape array-variable references.)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable</td>
<td>Variable that contains a string in which to preserve single-quotation marks.</td>
</tr>
</tbody>
</table>

**Usage**

This function is useful in SQL statements to defer evaluation of a variable reference until runtime. This prevents errors that result from the evaluation of a single-quote or apostrophe data character (for example, "Joe's Diner") as a delimiter. **Example A**: Consider this code:

```
<cfset mystring = "'Newton's Law', 'Fermat's Theorem'">
PreserveSingleQuotes(#mystring#) is
<cfoutput>
    #PreserveSingleQuotes(mystring)#
</cfoutput>
```

The output is as follows:

```
PreserveSingleQuotes(#mystring#) is 'Newton's Law', 'Fermat's Theorem'
```
**Example B**: Consider this code:

```coldfusion
<cfset list0 = " '1','2', '3' ">
<cfquery sql = "select * from foo where bar in (#list0#)"
```

ColdFusion escapes the single-quote characters in the list as follows:

```
""1"", ""2"", ""3"
```

The `cfquery` tag throws an error. You code this function correctly as follows:

```coldfusion
<cfquery sql = "select * from foo where bar in (#preserveSingleQuotes(list0)#)"
**tharwood 11/16
```

This function ensures that ColdFusion evaluates the code as follows:

```
'1', '2', '3'
```

**Example**

```coldfusion
<h3>PreserveSingleQuotes Example</h3><p>This is a useful function for creating lists of information to return from a query. In this example, we pick the list of Centers in Suisun, San Francisco, and San Diego, using the SQL grammar IN to modify a WHERE clause, rather than looping through the result set after the query is run.
<cfset List = "'Suisun', 'San Francisco', 'San Diego'">
<cfquery name = "GetCenters" datasource = "cfdocexamples">
  SELECT Name, Address1, Address2, City, Phone
  FROM Centers
  WHERE City IN (#PreserveSingleQuotes(List)#)
</cfquery>
<p>We found <cfoutput>#GetCenters.RecordCount#</cfoutput> records.
<cfoutput query = "GetCenters">
  #Name#<br>
  #Address1#
<cfif Address2 is not "">
    #Address2#
  </cfif>
  #City#<br>
  #Phone#
</cfoutput>
QueryAddColumn

Description

Adds a column to a query and populates its rows with the contents of a one-dimensional array. Pads query columns, if necessary, to ensure that all columns have the same number of rows.

Returns

The number of the column that was added.

Category

Query functions

Function syntax

```
QueryAddColumn(query, column-name [, datatype], array-name)
```

See also

`QueryNew`, `QueryAddRow`, `QuerySetCell`; `Managing data types for columns` in `Query of Queries user guide` in the `Developing ColdFusion Applications`

History

ColdFusion MX 7: Added the `datatype` parameter.

ColdFusion MX: Changed behavior: if a user attempts to add a column whose name is invalid, ColdFusion throws an error. (In earlier releases, ColdFusion permitted the add operation, but the user could not reference the column after adding it.)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>Name of a query object.</td>
</tr>
<tr>
<td>column-name</td>
<td>Name of the new column.</td>
</tr>
</tbody>
</table>
**datatype**

(Optional) Column data type. ColdFusion generates an error if data you add to the column is not of this type, or if it cannot convert the data to this type. The following data types are valid:

- Integer: 32-bit integer
- BigInt: 64-bit integer
- Double: 64-bit decimal number
- Decimal: Variable length decimal, as specified by java.math.BigDecimal
- VarChar: String
- Binary: Byte array
- Bit: Boolean (1=True, 0=False)
- Time: Time
- Date: Date (can include time information)

**array-name**

Name of an array whose elements populate the new column.

**Usage**

You can add columns to query objects, such as queries retrieved with the `cfquery` tag or queries created with the `queryNew` function. You cannot use the `QueryAddColumn` function on a cached query. This function is useful for generating a query object from the arrays of output parameters that Oracle stored procedures can generate. Adobe recommends that you use the optional `datatype` parameter. Without this parameter, ColdFusion must try to determine the column's data type when it uses the query object in a query of queries. Determining the data type requires additional processing, and can result in errors if ColdFusion does not guess the type correctly.

**Example**

The following example creates a query object, uses the `QueryAddColumn` function to add three columns to the object, and displays the results. Because two of the arrays that provide the data are shorter than the third, `QueryAddColumn` adds padding to the corresponding columns in the query.
<!--- Make a query. --->
<cfset myQuery = QueryNew("")>

<!--- Create an array. --->
<cfset FastFoodArray = ArrayNew(1)>
<cfset FastFoodArray[1] = "French Fries">
<cfset FastFoodArray[2] = "Hot Dogs">
<cfset FastFoodArray[3] = "Fried Clams">
<cfset FastFoodArray[4] = "Thick Shakes">

<!--- Use the array to add a column to the query. --->
<cfset nColumnNumber = QueryAddColumn(myQuery, "FastFood", "VarChar",
    FastFoodArray)>

<!--- Create a second array. --->
<cfset FineCuisineArray = ArrayNew(1)>
<cfset FineCuisineArray[1] = "Lobster">
<cfset FineCuisineArray[2] = "Flambe">

<!--- Use the array to add a second column to the query. --->
<cfset nColumnNumber2 = QueryAddColumn(myQuery, "FineCuisine", "VarChar",
    FineCuisineArray)>

<!--- Create a third array. --->
<cfset HealthFoodArray = ArrayNew(1)>
<cfset HealthFoodArray[1] = "Bean Curd">
<cfset HealthFoodArray[2] = "Yogurt">
<cfset HealthFoodArray[3] = "Tofu">

<!--- Use the array to add a third column to the query. --->
<cfset nColumnNumber3 = QueryAddColumn(myQuery, "HealthFood", "VarChar",
    HealthFoodArray)>

<!--- Display the results. --->
<table cellspacing = "2" cellpadding = "2" border = "0">
<tr>
<th align = "left">Fast Food</th>
<th align = "left">Fine Cuisine</th>
<th align = "left">Health Food</th>
</tr>
<cfoutput query = "myQuery">
<tr>
<td>#FastFood#</td>
<td>#FineCuisine#</td>
<td>#HealthFood#</td>
</tr>
</cfoutput>
</table>
QueryAddRow

Description

Adds a specified number of empty rows to a query.

Returns

The number of rows in the query

Category

Query functions

Function syntax

```
QueryAddRow(query [, number])
```

See also

QueryAddColumn, QuerySetCell, QueryNew; Creating a recordset with the QueryNew() function in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>Name of an executed query.</td>
</tr>
<tr>
<td>number</td>
<td>Number of rows to add to the query. The default value is 1.</td>
</tr>
</tbody>
</table>

Usage

Enhancements in ColdFusion 10 lets you specify a struct, an array of structs, or arrays with single or multiple dimensions to add rows to the query as shown in the following example:

```csharp
queryAddRow(myQuery1,
    [
        {id=2,name="Two"},
        {id=3,name="Three"},
        {id=4,name="Four"}
    ]);
queryAddRow(myQuery2,{id=4,name="Four"});
```
queryAddRow(myQuery1,
[1,"One"],
[2,"Two"],
[3,"Three"]
});

Example

<h3>QueryAddRow Example</h3>

<!--- start by making a query --->
<cfquery name="GetCourses" datasource="cfdocexamples">
SELECT Course_ID, Number, Descript
FROM Courses
</cfquery>

<p>The Query "GetCourses" has <cfoutput>#GetCourses.RecordCount#</cfoutput> rows.

<cfset CountVar = 0>
<cfloop CONDITION = "CountVar LT 15">
<cfset temp = QueryAddRow(GetCourses)>
<cfset CountVar = CountVar + 1>
<cfset Temp = QuerySetCell(GetCourses, "Number", 100*CountVar)>
<cfset Temp = QuerySetCell(GetCourses, "Descript", "Description of variable #CountVar#")>
</cfloop>

<p>After the QueryAddRow action, the query has
<cfoutput>#GetCourses.RecordCount#</cfoutput> records.
<cfoutput query="GetCourses">
<pre>#Course_ID# #Number# #Descript#</pre>
</cfoutput>
QueryNew

Description

Creates an empty query (query object).

Returns

An empty query with a set of named columns, or an empty query.

Category

Query functions

Function syntax

```
QueryNew(columnlist [, columntypelist[, rowData]])
```

History

ColdFusion MX 7: Added `columntypelist` parameter.

ColdFusion 10: Added `rowData` parameter.

See also

`QueryAddColumn`, `QueryAddRow`, `QuerySetCell`; Managing data types for columns in `Query of Queries` user guide in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnlist</td>
<td>Comma-delimited list of column names, or an empty string.</td>
</tr>
</tbody>
</table>
| columntypelist  | (Optional) Comma-delimited list specifying column data types. ColdFusion generates an error if the data you add to the column is not of this type, or if it cannot convert the data to this type. The following data types are valid:  
  - Integer: 32-bit integer  
  - BigInt: 64-bit integer  
  - Double: 64-bit decimal number  
  - Decimal: Variable length decimal, as specified by java.math.BigDecimal  
  - VarChar: String  
  - Binary: Byte array  
  - Bit: Boolean (1=True, 0=False)  
  - Time: Time  
  - Date: Date (can include time information) |
rowData

(Optional) Specifies data to add to the new query. Can be one of:

- A struct with key/value pairs representing a single row of data, where the key is the column name and the value is the data;
- an array of structs as per above, for adding multiple rows of data;
- an array of arrays where each inner array are values for a row of data, in the same order as the columns are specified in the columnlist parameter.

See "Usage" for examples.

Usage

If you specify an empty string in the columnlist parameter, use the QueryAddColumn function to add columns to the query. Adobe recommends that you use the optional columnlist parameter. Without this parameter, ColdFusion must try to determine data types when it uses the query object in a query of queries. Determining data types requires additional processing, and can result in errors if ColdFusion does not guess a type correctly. Enhancements in ColdFusion 10 lets you initialize the query data. You can specify a struct, an array of structs, or arrays with single or multiple dimensions to initialize the query as shown in the following example:

```plaintext
myQuery1 = queryNew("id,name","Integer, Varchar", {id=1,name="One"});
myQuery2 = queryNew("id,name","Integer, Varchar", [
    {id=1,name="One"},
    {id=2,name="Two"},
    {id=3,name="Three"}
]);

Example

The following example uses the QueryNew function to create an empty query with three columns. It populates two rows of the query and displays the contents of the query object and its metadata.
<!--- Create a new three-column query, specifying the column data types --->
<cfset myQuery = QueryNew("Name, Time, Advanced", "VarChar, Time, Bit")>

<!--- Make two rows in the query --->
<cfset QueryAddRow(MyQuery, 2)>

<!--- Set the values of the cells in the query --->
<cfset QuerySetCell(myQuery, "Name", "The Wonderful World of CMFL", 1)>
<cfset QuerySetCell(myQuery, "Time", "9:15 AM", 1)>
<cfset QuerySetCell(myQuery, "Advanced", False, 1)>
<cfset QuerySetCell(myQuery, "Name", "CFCs for Enterprise Applications", 2)>
<cfset QuerySetCell(myQuery, "Time", "12:15 PM", 2)>
<cfset QuerySetCell(myQuery, "Advanced", True, 2)>

<h4>The query object contents</h4>
<cfoutput query = "myQuery">
  #Name# #Time# #Advanced#
</cfoutput>
<br>
<br>
<h4>Using individual query data values</h4>
<cfoutput>
  #MyQuery.name[2]# is at #MyQuery.Time[2]#
</cfoutput>
<br>
<br>
<h4>The query metadata</h4>
<cfset querymetadata=getMetaData(myQuery)>
<cfdump var="#querymetadata#"
QueryConvertForGrid

Description

Converts query data to a structure that contains a paged subset of the query. Used in CFC functions that return data to Ajax format cfgrid controls in response to a bind expression.

Returns

A structure that contains one page of data from the query.

Category

Query functions

Function syntax

```
QueryConvertForGrid(query, page, pageSize)
```

See also

cfgrid, Dynamically filling form data in Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>Name of the query whose data is returned.</td>
</tr>
<tr>
<td>page</td>
<td>The specific page of query data to be returned. Pages are numbered starting with 1.</td>
</tr>
<tr>
<td>pageSize</td>
<td>Number of rows of query data on a page.</td>
</tr>
</tbody>
</table>

Usage

You can also create the return value for a cfgrid bind CFC without using this function if your query returns only a single grid page of data at a time. For more information see Using Ajax User Interface Components and Features in the Developing ColdFusion Applications.

Example

The following example shows how a CFC function that is called by an Ajax format cfgrid tag bind attribute, uses the QueryConvertForGrid function to prepare query data to return to the grid. The CFML page with the cfgrid tag has the following code:
The `getData` function in the `places.cfc` page has the following code:

```coldfusion
<cffunction name="getData" access="remote" output="false">
  <cfargument name="page"/>
  <cfargument name="pageSize"/>
  <cfargument name="gridsortcolumn"/>
  <cfargument name="gridstartdirection"/>
  <cfset query = "SELECT Emp_ID, FirstName, EMail FROM Employees">
  <cfif gridsortcolumn neq "" or gridstartdirection neq "">
    <cfset query=query & " order by #gridsortcolumn# #gridstartdirection#"/>
  </cfif>
  <cfquery name="team" datasource="cfdocexamples">
    <cfoutput>#query#</cfoutput>
  </cfquery>
  <cfreturn QueryConvertForGrid(team, page, pageSize)>
</cffunction>
```
**QuerySetCell**

**Description**

Sets a cell to a value. If no row number is specified, the cell on the last row is set. Starting with ColdFusion MX 7, you cannot add a string literal (for example, "All") to a column that is of type numeric, although this was allowed in previous versions of ColdFusion.

**Returns**

True, if successful; False, otherwise.

**Category**

Query functions

**Function syntax**

```
QuerySetCell(query, column_name, value [, row_number ])
```

**See also**

QueryAddColumn, QueryAddRow, QueryNew; Creating a recordset with the QueryNew() function in About recordsets in the Developing ColdFusion Applications

**History**

ColdFusion MX 7: Changed the behavior of the function so that it does type validation.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>Name of an executed query.</td>
</tr>
<tr>
<td>column_name</td>
<td>Name of a column in the query.</td>
</tr>
<tr>
<td>value</td>
<td>Value to set in the cell.</td>
</tr>
<tr>
<td>row_number</td>
<td>Row number. The default value is last row.</td>
</tr>
</tbody>
</table>

**Example**
<!--- This example shows the use of QueryAddRow and QuerySetCell --->

<!--- start by making a query --->
<cfquery name = "GetCourses" datasource = "cfdocexamples">
  SELECT Course_ID, Descript
  FROM Courses
</cfquery>

<p>The Query "GetCourses" has <cfoutput>#GetCourses.RecordCount#</cfoutput> rows.

<cfset CountVar = 0>
<cfloop CONDITION = "CountVar LT 15">
<cfset temp = QueryAddRow(GetCourses)>
<cfset CountVar = CountVar + 1>
<cfset Temp = QuerySetCell(GetCourses, "Number", 100*CountVar)>
<cfset CountVar = CountVar + 1>
<cfset Temp = QuerySetCell(GetCourses, "Descript", "Description of variable #CountVar#")>
</cfloop>

<p>After the QueryAddRow action, the query has <cfoutput>#GetCourses.RecordCount#</cfoutput> records.
<cfoutput query="GetCourses">
<pre>#{Course_ID# #Course_Number# #Descript#}</pre>
</cfoutput>
QuotedValueList

Description

Gets the values of each record returned from an executed query. ColdFusion does not evaluate the arguments.

Returns

A delimited list of the values of each record returned from an executed query. Each value is enclosed in single-quotiation marks.

Category

Query functions, List functions

Function syntax

```
QuotedValueList(query.column [, delimiter ])
```

See also

ValueList

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query.column</td>
<td>Name of an executed query and column. Separate query name and column name with a period.</td>
</tr>
<tr>
<td>delimiter</td>
<td>A string or a variable that contains one. Character(s) that separate column data.</td>
</tr>
</tbody>
</table>

Example
<cfset List = "'BIOL', 'CHEM'">  
<cfquery name = "GetDepartments" datasource = "cfdocexamples">  
  SELECT Dept_ID FROM Departments  
  WHERE Dept_ID IN (#PreserveSingleQuotes(List)#)  
</cfquery>  
<cfquery name = "GetCourseList" datasource = "cfdocexamples">  
  SELECT *  
  FROM CourseList  
  WHERE Dept_ID IN ('#GetDepartments.Dept_ID#')  
</cfquery>  
<cfoutput>  
List the course numbers that are in BIOL and CHEM (uses semicolon (;) as the delimiter):<br>  
#QuotedValueList(GetCourseList.CrNumber, ";")#<br>  </cfoutput>
Rand

Description
Generates a pseudo-random number.

Returns
A pseudo-random decimal number, in the range 0-1.

Category
Mathematical functions, Security functions

Function syntax
Rand([algorithm])

History
ColdFusion MX 7: Added the algorithm parameter.

See also
Randomize, RandRange

Parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| algorithm | (Optional) The algorithm to use to generate the random number. ColdFusion installs a cryptography library with the following algorithms:  
  - CFMX_COMPAT: the algorithm used in ColdFusion (default).  
  - SHA1PRNG: generates a number using the Sun Java SHA1PRNG algorithm. This algorithm provides greater randomness than the default algorithm  
  - IBMSecureRandom: for IBM WebSphere (IBM JVM does not support the SHA1PRNG algorithm). |

Usage
Call the Randomize function before calling this function to seed the random number generator. Seeding the generator ensures that the Rand function always generates the same sequence of pseudo-random numbers. This behavior is useful if you must reproduce a pattern consistently. ColdFusion MX uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section (except the default algorithm). The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

Example
The following example uses the SHA1PRNG algorithm to generate a single random number:

```
<h3>Rand Example</h3>
<cfoutput>
  <p>Rand("SHA1PRNG") returned: #Rand("SHA1PRNG")#</p>
  <p><A HREF="#CGI.SCRIPT_NAME">Try again</A></p>
</cfoutput>
```
Randomize

Description

Seeds the pseudo-random number generator with an integer number, ensuring repeatable number patterns.

Returns

A pseudo-random decimal number, in the range 0-1.

Category

Mathematical functions, Security functions

Function syntax

```
Randomize(number[, algorithm])
```

History

ColdFusion MX 7: Added the algorithm parameter.

See also

Rand, RandRange

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Integer number. If the number is not in the range -2,147,483,648 -2,147,483,647, ColdFusion generates an error.</td>
</tr>
<tr>
<td>algorithm</td>
<td>(Optional) The algorithm to use to generate the seed number. ColdFusion installs a cryptography library with the following algorithms:</td>
</tr>
<tr>
<td></td>
<td>• CFMX_COMPAT: the algorithm used in ColdFusion (default).</td>
</tr>
<tr>
<td></td>
<td>• SHA1PRNG: generates a number using the Sun Java SHA1PRNG algorithm. This algorithm provides greater randomness than the default algorithm. Note: due to a bug in ColdFusion 9, 10 and 11, this option does not work as prescribed. See bug 3792283 for details.</td>
</tr>
<tr>
<td></td>
<td>• IBMSecureRandom: for IBM WebSphere (IBM JVM does not support the SHA1PRNG algorithm).</td>
</tr>
</tbody>
</table>

Usage

Call this function before calling Rand to seed the random number generator. Seeding the generator ensures that the Rand function always generates the same sequence of pseudo-random numbers. This behavior is useful if you must
reproduce a pattern consistently. In Standard Edition, for all algorithms except the default algorithm, ColdFusion uses
the Java Cryptography Extension (JCE) and installs a Sun Java runtime that includes the Sun JCE default security
provider. This provider includes the algorithms listed in the Parameters section. The JCE framework includes
facilities for using other provider implementations; however, Adobe cannot provide technical support for third-party
security providers. In Enterprise Edition, ColdFusion also installs the RSA BSafe Crypto-J library. This provider adds
the following algorithms: FIPS186PRNG, MD5PRNG, DummyPRNG, OBFPNG. DummyPRNG always returns 0.

Example

The following example calls the Randomize function to seed the random number generator and generates 10
random numbers.

```
<cfset randomize(12345)> <!--- if one was to remove this line, the random numbers
are different every time --->
<cfloop index="i" from="1" to="10">
  <cfoutput>#rand()#</cfoutput><br>
</cfoutput><br>
```

RandRange

Description
Generates a pseudo-random integer in the range between two specified numbers.

Returns
A pseudo-random integer.

Category
Mathematical functions, Security functions

Function syntax
```
RandRange(number1, number2[, algorithm])
```

History
ColdFusion MX 7: Added the `algorithm` parameter.

See also
Rand, Randomize

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number1, number2</td>
<td>Integer numbers. If the numbers are not in the range -2,147,483,648 - 2,147,483,647, ColdFusion generates an error.</td>
</tr>
<tr>
<td>algorithm</td>
<td>(Optional) The algorithm to use to generate the random number. ColdFusion installs a cryptography library with the following algorithms:</td>
</tr>
<tr>
<td></td>
<td>• CFMX_COMPAT: the algorithm used in ColdFusion (default).</td>
</tr>
<tr>
<td></td>
<td>• SHA1PRNG: generates a number using the Sun Java SHA1PRNG algorithm. This algorithm provides greater randomness than the default algorithm</td>
</tr>
<tr>
<td></td>
<td>• IBMSecureRandom: for IBM WebSphere (IBM JVM does not support the SHA1PRNG algorithm.)</td>
</tr>
</tbody>
</table>

Usage
Very large positive or negative values for the `number1` and `number2` parameters might result in poor randomness in the results. To prevent this problem, do not specify numbers outside the range -1,000,000,000 - 1,000,000,000. ColdFusion uses the Java Cryptography Extension (JCE) and installs a Sun Java 1.4.2 runtime that
includes the Sun JCE default security provider. This provider includes the algorithms listed in the Parameters section (except the default algorithm). The JCE framework includes facilities for using other provider implementations; however, cannot provide technical support for third-party security providers.

Example

The following example contains a form that requires random number range values, and lets you optionally specify a random number seed value. It uses \texttt{cfform} controls and attributes to specify a default range, ensure that the range fields have values, and validate that the field values are in a specified integer range. When you submit the form, it checks whether the seed field has an empty string; if the field has a value, the code uses the number to seed the random number generator. It then generates and displays the random number.

```html
<h3>RandRange Example</h3>

```perl
<cfif IsDefined("Form.mySeed")>
  <!--- Do the following only if the form has been submitted. --->
  <cfif Form.mySeed NEQ ">
    <!--- Do the following only if the seed field has a non-empty string. --->
    <cfoutput>
      <b>Seed value is #FORM.mySeed#</b><br>
    </cfoutput>
    <br>
    <!--- Call Randomize to seed the random number generator. --->
    <cfset r = Randomize(FORM.mySeed, "SHA1PRNG")>
    <cfelse>
      <b>No Seed value submitted</b><br>
    </cfif>
  <!--- Generate and display the random number. --->
  <cfoutput>
    RandRange returned: #RandRange(FORM.myInt, FORM.myInt2, "SHA1PRNG")#  
  </cfoutput></p>
</cfif>

<!--- This form uses \texttt{cfform} input validation to check the input range. --->
<cfform action = "#CGI.SCRIPT_NAME#">
  <!--- Enter the random number Range: From --->
  <cfinput type = "Text" name = "MyInt" value = "1"
    RANGE = "-1000000000,1000000000"
    message = "Please enter a value between -1,000,000,000 and 1,000,000,000"
    validate = "integer" required = "Yes">
  To
  <cfinput type = "Text" name = "MyInt2" value = "9999"
    RANGE = "-1000000000,1000000000"
    message = "Please enter a value between --1,000,000,000and 1,000,000,000"
    validate = "integer" required = "Yes">
  <p>Enter a number to seed the randomizer:
    <cfinput type = "Text" name = "mySeed" RANGE = "-1000000000,1000000000"
      message = "Please enter a value between -1,000,000,000 and 1,000,000,000"
      validate = "integer" required = "No">
  <p><input type = "Submit" name = ">
</cfform>
```
ReEscape

Description

Takes a string and escapes characters that match regular expression control characters.

Returns

String appended with the escape characters

History

ColdFusion 10: Added this function

Syntax

reEscape(string____)

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string in which you have to escape characters that match regular expression characters.</td>
</tr>
</tbody>
</table>

Example

```cfc
<coutput>
#reescape("*.{}[]exam?ple")#
</coutput>
```
REFind

Description

Uses a regular expression (RE) to search a string for a pattern. The search is case sensitive. For more information on regular expressions, including escape sequences, anchors, and modifiers, see Using Regular Expressions in Functions in the Developing ColdFusion Applications.

Returns

Depends on the value of the returnsubexpressions parameter:

- If returnsubexpressions = "False":
  - The position in the string where the match begins
  - 0, if the regular expression is not matched in the string
- If returnsubexpressions = "True": a structure that contains two arrays, len and pos. The array elements are as follows:
  - If the regular expression is found in the string, the first element of the len and pos arrays contains the length and position, respectively, of the first match of the entire regular expression.
  - If the regular expression contains parentheses that group subexpressions, each subsequent array element contains the length and position, respectively, of the first occurrence of each group.
  - If the regular expression is not found in the string, the first element of the len and pos arrays contains 0.

Category

String functions

Function syntax

```coldfusion
REFind(reg_expression, string [, start, returnsubexpressions ] )
```

See also

Find, FindNoCase, REFindNoCase, REReplace, REReplaceNoCase

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg_expression</td>
<td>Regular expression for which to search. Case sensitive.</td>
</tr>
<tr>
<td>string</td>
<td>A string, or a variable that contains one, in which to search.</td>
</tr>
<tr>
<td>start</td>
<td>Optional. A positive integer, or a variable that contains one. Position in the string at which to start search. The default value is 1.</td>
</tr>
</tbody>
</table>
returnsubexpressions

Optional. Boolean. Whether to return substrings of reg_expression, in arrays named len and pos:

- True: if the regular expression is found, the first array element contains the length and position, respectively, of the first match. If the regular expression contains parentheses that group subexpressions, each subsequent array element contains the length and position, respectively, of the first occurrence of each group. If the regular expression is not found, the arrays each contain one element with the value 0.
- False: the function returns the position in the string where the match begins. Default.

Usage

This function finds the first occurrence of a regular expression in a string. To find the second and subsequent instances of the expression or of subexpressions in it, you call this function more than once, each time with a different start position. To determine the next start position, use the returnsubexpressions parameter, and add the value returned in the first element of the length array to the value in the first element of the position array.

Example

```cfml
<h3>REFind Example</h3>
<p>This example shows the use of the REFind function with and without the <i>returnsubexpressions</i> parameter set to True.
If you do not use the <i>returnsubexpressions</i> parameter, REFind returns the position of the first occurrence of a regular expression in a string starting from the specified position. Returns 0 if no occurrences are found.</p>

<cfoutput>
REFind("a+c", "abcaaccdd")
</cfoutput>

<cfoutput>
REFind("a+c", "abcaaccdd")
</cfoutput>

<cfoutput>
REFind("a+c", "abcaaccdd")
</cfoutput>

<cfoutput>
REFind("\[[:upper:\]\]", "abcaacCDD")
</cfoutput>

<cfoutput>
REFind("\[[?:&]\]rep = ", "report.cfm?rep = 1234&u = 5")
</cfoutput>

<!--- Set startPos to one; returnMatchedSubexpressions = TRUE --->

/hr size = "2" color = "#000OA0"
<p>If you use the <i>returnsubexpressions</i> parameter, REFind returns the position and length of the first occurrence of a regular expression in a string starting from the specified position. The position and length variables are stored in a structure. To access position and length information, use the keys <i>pos</i> and <i>len</i>, respectively.</p>
<cfset teststring = "The cat in the hat hat came back!">
<p>The string in which the function is to search is:
</cfoutput><cfoutput>
</cfoutput>.<p>
<p>The first call to REFind to search this string is:
</p><cfoutput>
</cfoutput>.<p>
<p>This function returns a structure that contains two arrays: pos and len.</p>
</cfoutput>
```
To create this structure you can use a CFSET statement, for example:

```
<CFSET st = REFind("[:alpha:]",testString,1,"TRUE")>
```

The number of elements in each array: #ArrayLen(st.pos)#.

The number of elements in the pos and len arrays is always one if you do not use parentheses in the regular expression.

The value of st.pos[1] is: #st.pos[1]#.
The value of st.len[1] is: #st.len[1]#.

Substring is #Mid(testString,st.pos[1],st.len[1])#.

However, if you use parentheses in the regular expression, the first element contains the position and length of the first instance of the whole expression. The position and length of the first instance of each parenthesized subexpression within is included in additional array elements.

For example:

```
<CFSET st1 = REFind("([[:alpha:]]+)\{1\},testString,1,"TRUE")>
```

The number of elements in each array is #ArrayLen(st1.pos)#.

First whole expression match; position is #st1.pos[1]#; length is #st1.len[1]#; whole expression match is #Mid(testString,st1.pos[1],st1.len[1])#.

Subsequent elements of the arrays provide the position and length of the first instance of each parenthesized subexpression therein.

```
<cfloop index = "i" from = "2" to = "#ArrayLen(st1.pos)#">
<cfoutput>Position is #st1.pos[i]#; Length is #st1.len[i]#; Substring is #Mid(testString,st1.pos[i],st1.len[i])#</cfoutput>
</cfloop>
```
</B></cfoutput></p>
</cfloop><br>
REFindNoCase

Description

Uses a regular expression (RE) to search a string for a pattern, starting from a specified position. The search is case-insensitive. For more information on regular expressions, including escape sequences, anchors, and modifiers, see Using Regular Expressions in Functions in the Developing ColdFusion Applications.

Returns

Depends on the value of the `returnsubexpressions` parameter:

- **If `returnsubexpressions` = "False":**
  - The position in the string where the match begins
  - 0, if the regular expression is not matched in the string
- **If `returnsubexpressions` = "True":** a structure that contains two arrays, `len` and `pos`. The array elements are as follows:
  - If the regular expression is found in the string, the first element of the `len` and `pos` arrays contains the length and position, respectively, of the first match of the entire regular expression.
  - If the regular expression contains parentheses that group subexpressions, each subsequent array element contains the length and position, respectively, of the first occurrence of each group.
  - If the regular expression is not found in the string, the first element of the `len` and `pos` arrays contains 0.

Category

String functions

Function syntax

```coldfusion
REFindNoCase(reg_expression, string [, start, returnsubexpressions])
```

See also

Find, FindNoCase, REFind, REReplace, REReplaceNoCase

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg_expression</td>
<td>Regular expression for which to search. Case-insensitive. For more information, see Using Regular Expressions in Functions in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String in which to search.</td>
</tr>
<tr>
<td>start</td>
<td>Optional. A positive integer or a variable that contains one. Position at which to start search. The default value is 1.</td>
</tr>
</tbody>
</table>
returnsubexpressions

Optional. Boolean. Whether to return substrings of reg_expression, in arrays named len and pos:

- True: if the regular expression is found, the first array element contains the length and position, respectively, of the first match. If the regular expression contains parentheses that group subexpressions, each subsequent array element contains the length and position, respectively, of the first occurrence of each group. If the regular expression is not found, the arrays each contain one element with the value 0.
- False: the function returns the position in the string where the match begins. Default.

Usage

This function finds the first occurrence of a regular expression in a string. To find the second and subsequent instances of the expression or of subexpressions in it, you call this function more than once, each time with a different start position. To determine the next start position, use the returnsubexpressions parameter, and add the value returned in the first element of the length array to the value in the first element of the position array.

Example

```cfset teststring = "The cat in the hat hat came back!"
<cfoutput>
  The string in which the function is to search is:
  #teststring#<b></cfoutput>
<cfset startPos = 1>
<cfset returnMatchedSubexpressions = True>
<cfset searchResult = REFindNoCase("[[:alpha:]]+", testString, startPos, "True")>
<cfoutput>
  The first call to REFindNoCase to search this string is:
  #REFindNoCase("[[:alpha:]]+", testString, 1, "True")#<b></cfoutput>
<cfoutput>
  This function returns a structure that contains two arrays: pos and len.<b></cfoutput>
<cfoutput>
  To create this structure you can use a CFSET statement,
```
for example:

```
<CFSET st = REFindNoCase("\[[[:alpha:]]]+\],testString,1,"True")>
<cfset st = REFindNoCase("\[[[:alpha:]]]+\],testString,1,"True")>
<p>
The number of elements in each array: #ArrayLen(st.pos)#.
</cfoutput></p>
<p>The number of elements in the pos and len arrays will always be one, if you do not use parentheses to denote subexpressions in the regular expression.</p>
<p>The value of st.pos[1] is: <cfoutput>#st.pos[1]#.</cfoutput></p>
<p>The value of st.len[1] is: <cfoutput>#st.len[1]#.</cfoutput></p>
<p>Substring is <b>#Mid(testString,st.pos[1],st.len[1])#</b>.
</cfoutput></p>
<hr size = "2" color = "#0000A0">
<p>However, if you use parentheses to denote subexpressions in the regular expression, the first element contains the position and length of the first instance of the whole expression. The position and length of the first instance of each subexpression within will be included in additional array elements.</p>
<p>For example:

```
<CFSET st1 = REFindNoCase("(\[[[:alpha:]]]+\]\(\[\])+(\1)",testString,1,"True")>
</CFSET>
<cfset st1 = REFindNoCase("(\[[[:alpha:]]]+\]\(\[\])+(\1)",testString,1,"True")>
<p>The number of elements in each array is #ArrayLen(st1.pos)#.
</cfoutput></p>
<p>First whole expression match; position is
</cfoutput>
#st1.pos[1]#; length is #st1.len[1]#;
whole expression match is <b>#Mid(testString,st1.pos[1],st1.len[1])#</b>.
</cfoutput></p>
<p>Subsequent elements of the arrays provide the position and length of the first instance of each parenthesized subexpression therein.</p>
```
<cfoutput></p>
</cfloop><br>
REMatch

Description

Uses a regular expression (RE) to search a string for a pattern, starting from a specified position. The search is case sensitive. For more information on regular expressions, including escape sequences, anchors, and modifiers, see Using Regular Expressions in Functions in the Developing ColdFusion Applications.

Returns

An array of strings that match the expression.

Category

String functions

Function syntax

REMatch(reg_expression, string)

See also

Find, FindNoCase, REFind, RERefind, REReplace, REReplaceNoCase, REMatchNoCase

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg_expression</td>
<td>Regular expression for which to search. Case sensitive. For more information, see Using Regular Expressions in Functions in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String in which to search.</td>
</tr>
</tbody>
</table>

Usage

This function finds all occurrence of a regular expression in a string.

Example

```cfs
<!--- Find all the URLs in a web page retrieved via cfhttp: -->
<!--- The search is case sensitive. --->
result = REMatch("https?://([-\w.]+)(:\d+)?/([\w/_\.]*)(\?S+)?" , cfhttp.filecontent);
```
**REMatchNoCase**

**Description**

Uses a regular expression (RE) to search a string for a pattern, starting from a specified position. The search is case-insensitive. For more information on regular expressions, including escape sequences, anchors, and modifiers, see **Using Regular Expressions in Functions** in the Developing ColdFusion Applications.

**Returns**

An array of strings that match the expression.

**Category**

**String functions**

**Function syntax**

```coldfusion
REMatchNoCase(reg_expression, string)
```

**See also**

*Find, FindNoCase, REFind, REReplace, REReplaceNoCase, REMatch*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg_expression</td>
<td>Regular expression for which to search. Case-insensitive. For more information, see Using Regular Expressions in Functions in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String in which to search.</td>
</tr>
</tbody>
</table>

**Example**

```coldfusion
<!--- Find all the URLs in a web page retrieved via cfhttp: --->
result = REMatch("https?://([-\w.]+)+(:\d+)?/([^\w/_\.]*(\?\S+)?)?", cfhttp.filecontent);
```
ReleaseComObject

Description

Releases a COM Object and frees up resources that it used.

Returns

Nothing.

Category

Extensibility functions

Function syntax

ReleaseComObject(objectName)

See also

CreateObject, cfobject

History

ColdFusion MX 6.1: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectName</td>
<td>Variable name of a COM object that was created using the CreateObject function or cfobject tag.</td>
</tr>
</tbody>
</table>

Usage

This function forcefully terminates and releases the specified COM object and all COM objects that it created. Use this function when the object is no longer in use, to quickly free up resources. If the COM object has a method, such as a quit method, that terminates the program, call this method before you call the ReleaseComObject function. This function can improve processing efficiency, but is not required for an application to work. If you do not use this function, the Java garbage collection mechanism eventually frees the resources. If you use this function on an object that is in use, the object is prematurely released and your application will get exceptions.

Example

```cfscript
<h3>ReleaseComObject Example</h3>
<cfscript>
obj = CreateObject("Com", "excel.application.9");
//code that uses the object goes here???.I'd like to fill this in with something???
obj.quit();
ReleaseComObject(obj);
</cfscript>
```
RemoveCachedQuery

Description

Removes the query with the details you provide from query cache.

Returns

Nothing

Syntax

removeCachedQuery(SQL_, datasource, params, region___)

History

ColdFusion 10: Added this function.

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>The query SQL.</td>
</tr>
<tr>
<td>datasource</td>
<td>The datasource you ran the query on.</td>
</tr>
<tr>
<td>params</td>
<td>(Optional) Array of parameter values passed to SQL.</td>
</tr>
<tr>
<td>region</td>
<td>(Optional) Specifies the cache region where you can place the cache object</td>
</tr>
</tbody>
</table>

Example

```cfc
<cfset sql = "SELECT * from art where artid = ?">
<cfquery name="q" datasource="cfartgallery" cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT * from art where artid = <cfqueryPARAM value = "1" CFSQLType = 'CF_SQL_INTEGER'>
</cfquery>
<cfset a = arrayNew(1)>
<cfset a[1] = 1>
<cfset removeCachedQuery(sql,"cfartgallery", a)>
```
**RemoveChars**

**Description**

Removes characters from a string.

**Returns**

A copy of the string, with count characters removed from the specified start position. If no characters are found, returns zero.

**Category**

String functions

**Function syntax**

```plaintext
RemoveChars(string, start, count)
```

**See also**

Insert, Len

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String in which to search.</td>
</tr>
<tr>
<td>start</td>
<td>A positive integer or a variable that contains one. Position at which to start search.</td>
</tr>
<tr>
<td>count</td>
<td>Number of characters to remove.</td>
</tr>
</tbody>
</table>

**Example**
<h3>RemoveChars Example</h3>
Returns a string with <i>count</i> characters removed from the start position. Returns 0 if no characters are found.

<cfif IsDefined("FORM.myString")>
  <cfif (FORM.numChars + FORM.start) GT Len(FORM.myString)>
    <p>Your string is only <cfoutput>#Len(FORM.myString)# characters long. Please enter a longer string, select fewer characters to remove or begin earlier in the string.</p>
  </cfifelse>
  <cfelse>
    <p>Your original string:  #FORM.myString#</p>
    <p>Your modified string: #RemoveChars(FORM.myString, FORM.start, FORM.numChars)#</p>
  </cfendif>
</cfiff>
RepeatString

Description

Creates a string that contains a specified number of repetitions of the specified string.

Returns

A string.

Category

String functions

Function syntax

```
RepeatString(string, count)
```

See also

CJustify, LJustify, RJustify

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one.</td>
</tr>
<tr>
<td>count</td>
<td>Number of repeats.</td>
</tr>
</tbody>
</table>

Example

```
<h3>RepeatString Example</h3>
<p>RepeatString returns a string created from <i>string</i>, repeated a specified number of times.</p>
<ul>
  <li>RepeatString("-", 10): <cfoutput>#RepeatString("-", 10)#</cfoutput></li>
  <li>RepeatString("&lt;BR&gt;", 3): <cfoutput>#RepeatString("<br>", 3)#</cfoutput></li>
  <li>RepeatString("", 5): <cfoutput>#RepeatString("", 5)#</cfoutput></li>
  <li>RepeatString("abc", 0): <cfoutput>#RepeatString("abc", 0)#</cfoutput></li>
  <li>RepeatString("Lorem Ipsum", 2):
    <cfoutput>#RepeatString("Lorem Ipsum", 2)#</cfoutput></li>
</ul>
```
Replace

Description

Replaces occurrences of substring1 in a string with substring2, in a specified scope. The search is case sensitive.

Returns

The string, after making replacements.

Category

String functions

Function syntax

Replace(string, substring1, substring2 \[, scope \])

See also

Find, REFind, ReplaceNoCase, ReplaceList, REReplace

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String in which to search.</td>
</tr>
<tr>
<td>substring1</td>
<td>A string or a variable that contains one. String for which to search</td>
</tr>
<tr>
<td>substring2</td>
<td>String that replaces substring1</td>
</tr>
<tr>
<td>scope</td>
<td>• one: replaces the first occurrence (default)</td>
</tr>
<tr>
<td></td>
<td>• all: replaces all occurrences</td>
</tr>
</tbody>
</table>

Usage

To remove a string, specify the empty string ("""") as substring2. You do not need to escape comma characters in strings. For example, the following code deletes the commas from the sentence:

```text
replace("The quick brown fox jumped over the lazy cow, dog, and cat.","","","All")
```
<h3>Replace Example</h3>

<p>The Replace function returns <i>string</i> with <i>substring1</i> replaced by <i>substring2</i> in the specified scope. This is a case-sensitive search.</p>

<cfif IsDefined("FORM.MyString")>
<p>Your original string, <cfoutput>#FORM.MyString#</cfoutput></p>
<p>You wanted to replace the substring <cfoutput>#FORM.MySubstring1#</cfoutput> with the substring <cfoutput>#FORM.MySubstring2#</cfoutput>.</p>
<p>The result: <cfoutput>#Replace(FORM.myString, FORM.MySubstring1, FORM.mySubString2)#</cfoutput></p>
</cfif>
ReplaceList

Description
Replaces occurrences of the elements from a delimited list in a string with corresponding elements from another delimited list. The search is case sensitive.

Returns
A copy of the string, after making replacements.

Category
List functions, String functions

Function syntax

<table>
<thead>
<tr>
<th>Function Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplaceList(string, list1, list2)</td>
<td></td>
</tr>
<tr>
<td>ReplaceList(string, list1, list2, delimiter)</td>
<td></td>
</tr>
<tr>
<td>ReplaceList(string, list1, list2, delimiter_list1, delimiter_list2)</td>
<td></td>
</tr>
</tbody>
</table>

See also
Find, REFind, Replace, REReplace

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string, or a variable that contains one, within which to replace substring</td>
</tr>
<tr>
<td>list1</td>
<td>Comma-delimited list of substrings for which to search</td>
</tr>
<tr>
<td>list2</td>
<td>Comma-delimited list of replacement substrings</td>
</tr>
<tr>
<td>delimiter</td>
<td>Common delimiter for both search and replacement.</td>
</tr>
<tr>
<td>delimiter_list1</td>
<td>Delimiter for search.</td>
</tr>
<tr>
<td>delimiter_list2</td>
<td>Delimiter for replacement.</td>
</tr>
</tbody>
</table>

Usage
The list of substrings to replace is processed sequentially. If a list1 element is contained in list2 elements, recursive replacement might occur. The second example shows this.

Example 1
The ReplaceList function returns `<I>string</I>` with `<I>substringlist1</I>` (e.g. "a,b") replaced by `<I>substringlist2</I>` (e.g. "c,d") in the specified scope.

```cfif IsDefined("FORM.MyString")>
<p>Your original string, <cfoutput>FORM.MyString</cfoutput>
<p>You wanted to replace the substring <cfoutput>FORM.MySubstring1</cfoutput> with the substring <cfoutput>FORM.MySubstring2</cfoutput>.
<p>The result: <cfoutput>Replacelist(FORM.myString, FORM.MySubstring1, FORM.mySubString2)</cfoutput>
</cfif>
<form action = "replacelist.cfm" method="post">
<p>String 1<br><input type = "Text" value = "My Test String" name = "MyString">
<p>Substring 1 (find this list of substrings)<br><input type = "Text" value = "Test, String" name = "MySubstring1">
<p>Substring 2 (replace with this list of substrings)<br><input type = "Text" value = "Replaced, Sentence" name = "MySubstring2">
<p><input type = "Submit" value = "Replace and display" name = "" />
</form>

<h3>Replacelist Example Two</h3>
<cfset stringtoreplace = "The quick brown fox jumped over the lazy dog.">
<cfoutput>
#ReplaceList(stringtoreplace,"dog,brown,fox,black", "cow,black,ferret,white")#
</cfoutput>

Example 2

In the following example, the delimiter applies to both the lists:

```<h3>Replacelist Example One</h3>
<cfset stringtoreplace = "The quick brown fox jumped over the lazy dog.">
<cfoutput>
</cfoutput></code>

Example 3

In the following example, delimiter is specific to individual lists:

```<h3>Replacelist Example Two</h3>
<cfset stringtoreplace = "The quick brown fox jumped over the lazy dog.">
<cfoutput>
</cfoutput>```
ReplaceNoCase

Description

Replaces occurrences of `substring1` with `substring2`, in the specified scope. The search is case-insensitive.

Returns

A copy of the string, after making replacements.

Category

String functions

Function syntax

```
ReplaceNoCase(string, substring1, substring2 [, scope ])
```

See also

Find, REFind, Replace, ReplaceList, REReplace

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string (or variable that contains one) within which to replace substring.</td>
</tr>
<tr>
<td>substring1</td>
<td>String (or variable that contains one) to replace, if found.</td>
</tr>
<tr>
<td>substring2</td>
<td>String (or variable that contains one) that replaces <code>substring1</code>.</td>
</tr>
</tbody>
</table>
| scope     | • one: replaces the first occurrence (default).  
            • all: replaces all occurrences. |

Example
ReplaceNoCase Example

The ReplaceNoCase function returns <I>string</I> with <I>substring1</I> replaced by <I>substring2</I> in the specified scope. The search/replace is case-insensitive.

<cfif IsDefined("FORM.MyString")>
Your original string, <cfoutput>#FORM.MyString#</cfoutput>
You wanted to replace the substring <cfoutput>#FORM.MySubString1#</cfoutput> with the substring <cfoutput>#FORM.MySubString2#</cfoutput>.
The result: <cfoutput>#ReplaceNoCase(FORM.myString, FORM.MySubString1, FORM.mySubString2)#</cfoutput>
</cfif>
REReplace

Description

Uses a regular expression (RE) to search a string for a string pattern and replace it with another. The search is case sensitive.

Returns

If the scope parameter is set to one, returns a string with the first occurrence of the regular expression replaced by the value of substring. If the scope parameter is set to all, returns a string with all occurrences of the regular expression replaced by the value of substring. If the function finds no matches, it returns a copy of the string unchanged.

Category

String functions

Function syntax

```
REReplace(string, reg_expression, substring [, scope ])
```

See also

REFind, Replace, ReplaceList, REReplaceNoCase

History

ColdFusion MX: Added supports for the following special codes in a replacement substring, to control case conversion:

- \u - uppercase the next character
- \l - lowercase the next character
- \U - uppercase until \E
- \L - lowercase until \E
- \E - end \U or \L

For more information on new features, see REFind.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one. String within which to search.</td>
</tr>
<tr>
<td>reg_expression</td>
<td>Regular expression to replace. The search is case sensitive.</td>
</tr>
<tr>
<td>substring</td>
<td>A string or a variable that contains one. Replaces reg_expression.</td>
</tr>
</tbody>
</table>
### Usage

For details on using regular expressions, see [Using Regular Expressions in Functions](#) in the [Developing ColdFusion Applications](#).

### Example

```cfc
ten The REReplace function returns `<i>string</i>` with a regular expression replaced with `<i>substring</i>` in the specified scope. Case-sensitive search.
ten REReplace("CABARET","C|B","G","ALL"):  
<cfoutput>#REReplace("CABARET","C|B","G","ALL")#</cfoutput>  
ten REReplace("CABARET","[A-Z]","G","ALL"):  
<cfoutput>#REReplace("CABARET","[A-Z]","G","ALL")#</cfoutput>  
ten REReplace("I love jellies","jell(y|ies)","cookies"):  
<cfoutput>#REReplace("I love jellies","jell(y|ies)","cookies")#</cfoutput>  
ten REReplace("I love jelly","jell(y|ies)","cookies"):  
<cfoutput>#REReplace("I love jelly","jell(y|ies)","cookies")#</cfoutput>
```
## REReplaceNoCase

### Description

Uses a regular expression to search a string for a string pattern and replace it with another. The search is case-insensitive.

### Returns

- If `scope = "one"`: returns a string with the first occurrence of the regular expression replaced by the value of `substring`.
- If `scope = "all"`: returns a string with all occurrences of the regular expression replaced by the value of `substring`.
- If the function finds no matches: returns a copy of the string, unchanged.

### Category

String functions

### Function syntax

```
REReplaceNoCase(string, reg_expression, substring [, scope ])
```

### See also

REFind, RERefindNoCase, Replace, ReplaceList

### History

ColdFusion MX: Changed behavior: this function inserts the following special characters in regular expression replacement strings, to control case conversion: \u, \U, \l, \L, and \E. If any of these strings is present in a ColdFusion 5 application, insert a backslash before it (for example, change "\u" to "\u").

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one.</td>
</tr>
<tr>
<td>reg_expression</td>
<td>Regular expression to replace. For more information, see Using Regular Expressions in Functions in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>substring</td>
<td>A string or a variable that contains one. Replaces <code>reg_expression</code>.</td>
</tr>
<tr>
<td>scope</td>
<td>• one: replaces the first occurrence of the regular expression. Default.</td>
</tr>
<tr>
<td></td>
<td>• all: replaces all occurrences of the regular expression.</td>
</tr>
</tbody>
</table>
Usage

For details on using regular expressions, see Using Regular Expressions in Functions in the Developing ColdFusion Applications.

Example

```coldfusion
<p>The REReplaceNoCase function returns <i>string</i> with a regular expression replaced with <i>substring</i> in the specified scope. This is a case-insensitive search.
</p>
<p>REReplaceNoCase("cabaret","C|B","G","ALL"): <cfoutput>$REReplaceNoCase("cabaret","C|B","G","ALL")$</cfoutput></p>
<p>REReplaceNoCase("cabaret","[A-Z]","G","ALL"): <cfoutput>$REReplaceNoCase("cabaret","[A-Z]","G","ALL")$</cfoutput></p>
<p>REReplaceNoCase("I LOVE JELLIES","jell(y|ies)","cookies"): <cfoutput>$REReplaceNoCase("I LOVE JELLIES","jell(y|ies)","cookies")$</cfoutput></p>
<p>REReplaceNoCase("I LOVE JELLY","jell(y|ies)","cookies"): <cfoutput>$REReplaceNoCase("I LOVE JELLY","jell(y|ies)","cookies")$</cfoutput>
```
RestDeleteApplication

Description

Unregisters the directory path if it is already registered.

Returns

Nothing

History

ColdFusion 10: Added this function

Syntax

RestDeleteApplication("dirPath")

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirPath</td>
<td>Required. Path to the directory to be unregistered. If the path is not valid, it results in an error.</td>
</tr>
</tbody>
</table>

Example

```cfset RestDeleteApplication("C:/ColdFusion10/cfusion/wwwroot/restexample/")```
RestSetResponse

Description

Sets the custom responses.

History

ColdFusion 10: Added this function

Syntax

restSetResponse(response)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>A struct that contains the response details.</td>
</tr>
</tbody>
</table>

Warning

restSetResponse() can only be used on function with a return type of void. It will cause errors if it is used on functions that have any other return type.

Example

```xml
<cffunction name="create" httpMethod="POST" produces="application/xml">
  <cfargument name="id" type="numeric">
    <cfargument name="name" type="String">
      <cfset var response={
        status=201,
        content="<customer id="&id&" name="&name&"/>
      },
      headers={
      }
    }
    <cfset restSetResponse(response)>
  </cffunction>
```
RestInitApplication

Description

Registers the directory path with the service mapping provided. If no service mapping is provided, the application name is used. If the rest application is already registered, it is refreshed.

Returns

Nothing

History

ColdFusion 11: Added the options attribute.

ColdFusion 10: Added this function

Syntax

RestInitApplication("dirPath" [, "serviceMapping", "options"] )

Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirPath</td>
<td>Required. Path to the directory to be registered.</td>
</tr>
<tr>
<td>serviceMapping</td>
<td>Optional. Alternate string to be used for application name while calling the REST service.</td>
</tr>
<tr>
<td>options</td>
<td>Optional. Additional options passed for initializing the application.</td>
</tr>
<tr>
<td></td>
<td>• &quot;host&quot; - The application will be registered for the particular host.</td>
</tr>
<tr>
<td></td>
<td>• &quot;useHost&quot; - true/false - The application will extract the host from the URL and it will be used as the host name.</td>
</tr>
<tr>
<td></td>
<td>• &quot;isdefault&quot; - Specifies whether the application should be registered as a default application or not. For accessing a default application, the URL should not contain the application name.</td>
</tr>
</tbody>
</table>

Example

```cset
<cfset RestInitApplication("C:/ColdFusion10/cfusion/wwwroot/restexample/", "restexample")>
```
Reverse

Description

Reverses the order of items, such as the characters in a string or the digits in a number.

Returns

A copy of string, with the characters in reverse order.

Category

String functions

Function syntax

Reverse(string)

See also

Left, Mid, Right

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

Usage

You can call this function on a number with code such as the following:

```
<cfoutput>reverse(6*2) equals #reverse(6*2)#<cfoutput>
```

This code outputs the following:

```
reverse(6*2) equals 21
```

Example
<h3>Reverse Example</h3>
<p>Reverse returns your string with the positions of the characters reversed.
<cfif IsDefined("FORM.myString")>
  <cfif FORM.myString is not ">
    <p>Reverse returned:
    <cfoutput>#Reverse(FORM.myString)#</cfoutput>
  </cfelse>
  <p>Please enter a string to be reversed.
</cfif>
</cfif>
<form action = "reverse.cfm">
<p>Enter a string to be reversed:
<input type = "Text" name = "MyString">
<p><input type = "Submit" name = ">
</form>
Right

Description

Gets a specified number of characters from a string, beginning at the right. Returns the specified number of characters from the end (or right side) of the specified string.

Returns

- If the length of the string is greater than or equal to `count`, the rightmost `count` characters of the string
- If `count` is greater than the length of the string, the whole string
- If `count` is greater than 1, and the string is empty, an empty string

Category

String functions

Function syntax

```
Right(string, count)
```

See also

`Left`, `Mid`, `Reverse`

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one.</td>
</tr>
<tr>
<td>count</td>
<td>A positive integer that specifies the maximum number of characters to return.</td>
</tr>
</tbody>
</table>

Example
<!-- Simple Right Example--->
<cfoutput>
#Right("See the quick red fox jump over the fence", 9)#
<br>
#Right("ColdFusion", 6)#
</cfoutput>

<!-- Right Example using form input --->
<h3>Right Example</h3>
<cfif IsDefined("Form.MyText")>
<!-- If len returns 0 (zero), then show error message. --->
<cfif Len(FORM.myText)>
<cfif Len(FORM.myText) LTE FORM.removeChars>
<cfoutput><p style="color: red; font-weight: bold">Your string #FORM.myText# only has #Len(FORM.myText)# characters. You cannot output the #FORM.removeChars# rightmost characters of this string because it is not long enough.</p></cfoutput>
<cfelse>
<cfoutput><p>Your original string: <strong>#FORM.myText#</strong>
Your changed string, showing only the <strong>#FORM.removeChars#</strong> rightmost characters:
<strong>#right(Form.myText, FORM.removeChars)#</strong></p></cfoutput>
</cfif>
</cfif>
<cfelse>
<p style="color: red; font-weight: bold">Please enter a string of more than 0 (zero) characters.</p>
</cfif>
</cfif>
<form action="<cfoutput>#CGI.ScriptName#</cfoutput>" method="POST">
<p>Type in some text<br />
<input type="Text" name="myText"></p>
<p>How many characters from the right do you want to show?
<select name="RemoveChars">
<option value="1">1
<option value="3" selected>3
<option value="5">5
<option value="7">7
<option value="9">9</select>
<input type="Submit" name="Submit" value="Remove characters"></p>
</form>
RJustify

Description

Right justifies characters of a string.

Returns

A copy of a string, right-justified in the specified field length.

Category

Display and formatting functions, String functions

Function syntax

RJustify(string, length)

See also

CJustify, LJustify

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string enclosed in quotation marks, or a variable that contains one.</td>
</tr>
<tr>
<td>length</td>
<td>A positive integer or a variable that contains one. Length of field in which to justify string.</td>
</tr>
</tbody>
</table>

Example
<!-- This example shows how to use RJustify -->
<cfparam name = "jstring" default = "">

<cfif IsDefined("FORM.justifyString")>
    <cfset jstring = rjustify(FORM.justifyString, 35)>
</cfif>
<html>
<head>
<title>RJustify Example</title>
</head>
<body>
<h3>RJustify Function</h3>
<p>Enter a string. It will be right justified within the sample field</p>
<form action = "rjustify.cfm">
    <p><input type = "Text" value = "#jString#" size = 35 name = "justifyString"></p>
    <p><input type = "Submit" name = ""> <input type = "reset"></p>
</form>
### Round

**Description**

Rounds a number to the closest integer.

**Returns**

An integer.

**Category**

Mathematical functions

**Function syntax**

```
Round(number)
```

**See also**

`Ceiling`, `Fix`, `Int`

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Number to round</td>
</tr>
</tbody>
</table>

**Usage**

Use this function to round a number. This function rounds numbers that end with .5 up to the nearest integer. It rounds 3.5 to 4 and -3.5 to -3.

**Example**

```html
<h3>Round Example</h3>
<p>This function rounds a number to the closest integer.</p>
<ul>
  <li>Round(7.49) : <cfoutput>#Round(7.49)#</cfoutput></li>
  <li>Round(7.5) : <cfoutput>#Round(7.5)#</cfoutput></li>
  <li>Round(-10.775) : <cfoutput>#Round(-10.775)#</cfoutput></li>
  <li>Round(-35.5) : <cfoutput>#Round(-35.5)#</cfoutput></li>
  <li>Round(35.5) : <cfoutput>#Round(35.5)#</cfoutput></li>
  <li>Round(1.2345*100)/100 : <cfoutput>#Round(1.2345*100)/100#</cfoutput></li>
</ul>
```
RTrim

Description

Removes spaces from the end of a string.

Returns

A copy of string, after removing trailing spaces.

Category

String functions

Function syntax

RTrim(string)

See also

LTrim, Trim

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

Example

```html
<h3>RTrim Example</h3>

<cfif IsDefined("FORM.myText")>
<cfoutput>
<pre>
Your string: "#FORM.myText#"
Your string: "#RTrim(FORM.myText)#"
(right trimmed)
</pre>
</cfoutput>
</cfif>

<form action = "Rtrim.cfm" method="post">
<p>Enter some text. It will be modified by Rtrim to remove spaces from the right.
<p><input type = "Text" name = "myText" value = "TEST ">
<p><input type = "Submit" name = "">
</form>
```

#back to top
QueryGetRow

A new function, QueryGetRow, has been introduced in ColdFusion 11. This function returns a struct having all the columns as keys and their corresponding values:

```coldfusion
row_struct = QueryGetRow(queryName, row_index);

See the following example:
<cfquery name="myQuery" result="myresult" datasource="artGallery">
    select * from art where ARTID > 3
</cfquery>
<cfset firstRow = QueryGetRow(myQuery, 1) >
<cfdump var="#firstRow#" >
```

Example

```coldfusion
<cfquery name="myQuery" result="myresult" datasource="artGallery" fetchclientinfo="yes" >
    select * from art where ARTID >
    <cfqueryparam value="2" cfsqltype="CF_SQL_INTEGER">
</cfquery>
<cfdump var="#myQuery#" >
<cfset data = QueryGetRow(myQuery, 1) >
<cfdump var="#data#" >
```
QueryExecute

Description

Apart from the generic script syntax for the tag, there is another script alternative to cfquery tag, provided by the QueryExecute function. This function simplifies the query execution in the CFScript block.

The usage:

```coldfusion
QueryExecute(sql_str, queryParams, queryOptions);
```

See the following example:

```coldfusion
QueryExecute("select * from Employees where empid=1");

QueryExecute("select * from Employee where country=:country and citizenship=:country", {country='USA'});

QueryExecute("select * from Employee where country=:country and citizenship=:country", {country={value='USA', CFSQLType='CF_SQL_CLOB', list=true, otherqueryParam_attributes}, });

QueryExecute("select * from Employee where country=:country and citizenship=:country", {country='USA'}, {datasource="cfartgallery", cachename="employees"})

QueryExecute ("select * from Artists where artistid=? and country=?", [1, "USA"], {datasource="cfartgallery"});
```

Note the function:

```coldfusion
QueryExecute(sql [,a struct or array of struct for all named parameters ]);
```

This function lets you pass unnamed parameters to the query. Use '?' (question mark) as the place-holder for the parameters. The values to the parameters should be passed as an array to `params`.

For more information on using the unnamed parameters, see this document.

Note that the following points are applicable:

- If there are any Query parameters, they must be passed as the second argument of the function. It can be either a struct or an array.
- If you need to provide additional attribute as a Query parameter (like CFSQLType, list, and separator), you can pass these as struct against that column name.
- The Query and datasource properties will be passed as the third argument.

The Query tag is also available as a ColdFusion function apart from the generic script syntax for the CFQuery tag. This function simplifies the query execution in the CFScript block.
Usage
QueryExecute(sql_str, queryParams, queryOptions);

History
ColdFusion 11: Added this function

Example 1
<cfset qoptions = {result="myresult", datasource="artGallery", fetchclientinfo="yes"}>

<cfset myquery = QueryExecute("select * from art where ARTID < 5", [], qoptions)>
<cfdump var="#myQuery#" >

<cfset myquery1 = QueryExecute("select * from art where ARTID < ?", [4], qoptions)>
<cfdump var="#myQuery1#" >

<cfset myquery2 = QueryExecute("select * from art where ARTID < :artid and artistid=:aid ",
{artid={value=100}, aid=2}, qoptions)>
<cfdump var="#myQuery2#" >

Example 2

QueryExecute ("select from Employees where empid=1");
QueryExecute("select from Employee where country=:country and citizenship=:country",
{country='USA'});
QueryExecute("select from Employee where country=:country and citizenship=:country",
{country={value='USA', CFSQLType='CF_SQL_CLOB', list=true, otherqueryParam_attributes}, });
QueryExecute("select from Employee where country=:country and citizenship=:country",
{country='USA'},
{datasource="cfartgallery", cachename="employees"})
QueryExecute ("select from Artists where artistid=? and country=?", [1, "USA"],
{datasource="cfartgallery"});

Note the function:
QueryExecute(sql [,a struct or array of struct for all named parameters ]);

This function lets you pass unnamed parameters to the query. Use '?' (question mark) as the place-holder for the parameters. The values to the parameters should be passed as an array to params.

For more information on using the unnamed parameters, see this document.
Note that the following points are applicable:

- If there are any Query parameters, they must be passed as the second argument of the function. It can be either a struct or an array.
- If you need to provide additional attribute as a Query parameter (like CFSIZEType, list, and separator), you can pass these as struct against that column name.
- The Query and datasource properties will be passes as the third argument.
- You can write the SQL statement with a positional parameter.
Functions s

- Second
- SendGatewayMessage
- Serialize
- SerializeJSON
- SerializeXML
- SessionGetMetaData
- SessionInvalidate
- SessionRotate
- SetEncoding
- SetLocale
- SetProfileString
- SetVariable
- Sgn
- Sin
- Sleep
- SpanExcluding
- SpanIncluding
- SpreadsheetAddAutofilter
- SpreadsheetAddColumn
- SpreadsheetAddFreezePane
- SpreadsheetAddImage
- SpreadsheetAddInfo
- SpreadsheetAddPagebreaks
- SpreadsheetAddRow
- SpreadsheetAddRows
- SpreadsheetAddSplitPane
- SpreadsheetCreateSheet
- SpreadsheetDeleteColumn
- SpreadsheetDeleteColumns
- SpreadsheetDeleteRow
- SpreadsheetDeleteRows
- SpreadsheetFormatCell
- SpreadsheetFormatCellRange
- SpreadsheetFormatColumn
- SpreadsheetFormatColumns
- SpreadsheetFormatRow
- SpreadsheetFormatRows
- SpreadsheetGetCellComment
- SpreadsheetGetCellFormula
- SpreadsheetGetCellValue
- SpreadsheetInfo
- SpreadsheetMergeCells
- SpreadsheetNew
- SpreadsheetRead
- SpreadsheetReadBinary
- SpreadsheetRemoveSheet
- SpreadsheetSetActiveSheet
- SpreadsheetSetActiveSheetNumber
- SpreadsheetSetCellComment
- SpreadsheetSetCellFormula
- SpreadsheetSetCellValue
- SpreadsheetSetColumnWidth
- SpreadsheetSetFooter
- SpreadsheetSetHeader
- SpreadsheetSetRowHeight
- SpreadsheetShiftColumns
- SpreadsheetShiftRows
- SpreadsheetWrite
- Sqr
- StripCR
- StructAppend
- StructClear
- StructCopy
- StructCount
- StructDelete
- StructEach
- StructFilter
- StructFind
- StructFindKey
- StructFindValue
- StructGet
- StructInsert
- StructIsEmpty
- StructKeyArray
- StructKeyExists
- StructKeyList
- StructMap
- StructNew
- StructReduce
- StructSort
- StructUpdate
**Second**

**Description**

Extracts the ordinal for the second from a date/time object.

**Returns**

An integer in the range 0-59.

**Category**

*Date and time functions*

**Function syntax**

```
Second(date)
```

**See also**

*DatePart, Hash, Minute*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date/time object</td>
</tr>
</tbody>
</table>

**Usage**

When passing a date/time object as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object.

**Example**

```cftags
<!--- This example shows the use of Hour, Minute, and Second --->
<h3>Second Example</h3>
<cfoutput>
   The time is currently #TimeFormat(Now())#.  
   We are in hour #Hour(Now())#, Minute #Minute(Now())#  
   and Second #Second(Now())# of the day.
</cfoutput>
```
SendGatewayMessage

Description

Sends an outgoing message through a ColdFusion event gateway.

Returns

String. The value returned depends on the gateway type.

Category

Extensibility functions

Function syntax

SendGatewayMessage(gatewayID, data)

See also

GetGatewayHelper, IM gateway message sending commands, SMS Gateway CFEvent structure and commands, CFML event gateway SendGatewayMessage data parameter, and Sending a message using the SendGatewayMessage function in Developing an event gateway application in the Developing ColdFusion Applications

History

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayID</td>
<td>Identifier of the gateway to send the message. Must be the Gateway ID of one of the ColdFusion event gateway instances configured on the ColdFusion Administrator Event Gateways section's Gateways page.</td>
</tr>
<tr>
<td>data</td>
<td>A ColdFusion structure. The contents of the structure depend on the event gateway type, but typically include a MESSAGE field that contains the message to send and a field that contains the destination address.</td>
</tr>
</tbody>
</table>

Usage

The SendGatewayMessage function calls the specified gateway's outgoingMessage method. The value returned by the function depends on the gateway type. The following table describes the return values for standard ColdFusion gateway types:

<table>
<thead>
<tr>
<th>Gateway type</th>
<th>Return values</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Gateway Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous CFML</td>
<td>If the message was queued for delivery to the CFC, returns True; False, otherwise.</td>
</tr>
<tr>
<td>Lotus SameTime</td>
<td>If the message or command was successful, returns OK. If an error occurred, returns a string indicating the cause.</td>
</tr>
<tr>
<td>SMS</td>
<td>If the gateway is in asynchronous mode, returns the empty string immediately. If the gateway is in synchronous mode, the function waits for the gateway to return a response. If the message was successfully sent to the short message service center (SMSC), returns the message ID from the SMSC. If an error occurred, returns a string indicating the cause.</td>
</tr>
<tr>
<td>XMPP</td>
<td>If the message or command was successful, returns OK. If an error occurred, returns a string indicating the cause.</td>
</tr>
</tbody>
</table>

Example

The following example uses an instance of the CFML gateway to log messages asynchronously to a file. To use this example, configure an instance of the CFML gateway with the name “Asynch Logger” in the ColdFusion Administrator. This gateway instance must use a CFC that takes the message and logs it. For sample CFC code, see [Using the CFML event gateway for asynchronous CFCs](#) in the Developing ColdFusion Applications.

```cfs
tooltip
Sending an event to the CFML event gateway that is registered in the ColdFusion Administrator as Asynch Logger.<br>
<cfs>
  status = false;
  props = structNew();
  props.message = "Replace me with a variable with data to log";
  status = SendGatewayMessage("Asynch Logger", props);
  if (status IS True) WriteOutput(Event Message "#props.message#" has been sent.);
</cfs>
```
SerializeJSON

Description

Converts ColdFusion data into a JSON (JavaScript Object Notation) representation of the data.

Returns

A string that contains a JSON representation of the parameter value.

Category

Conversion functions

Syntax

```
SerializeJSON(var [, serializeQueryByColumns, useCustomSerializer])
```

See also


History

ColdFusion 11: Added the attribute. useCustomSerializer.

ColdFusion 8: Added function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>A ColdFusion data value or variable that represents one.</td>
</tr>
<tr>
<td>serializeQueryByColumns</td>
<td>A Boolean value that specifies how to serialize ColdFusion queries.</td>
</tr>
<tr>
<td></td>
<td>• false (the default): Creates an object with two entries: an array of column names and an array of row arrays. This format is required by the HTML format cgrid tag.</td>
</tr>
<tr>
<td></td>
<td>• true: Creates an object that corresponds to WDDX query format. For more information, see the Usage section.</td>
</tr>
<tr>
<td>useCustomSerializer</td>
<td>true/false. Whether to use the customSerializer or not. The default value is true. Note that the custom serialize will always be used for serialization. If false, the JSON serialization will be done using the default ColdFusion behavior.</td>
</tr>
</tbody>
</table>
Usage

This function is useful for generating JSON format data to be consumed by an Ajax application. The `SerializeJSON` function converts ColdFusion dates and times into strings that can be easily parsed by the JavaScript `Date` object. The strings have the following format:

```
MonthName, DayNumber Year Hours:Minutes:Seconds
```

The `SerializeJSON` function converts the ColdFusion date time object for October 3, 2007 at 3:01 PM, for example, into the JSON string "October, 03 2007 15:01:00". The `SerializeJSON` function with a `false serializeQueryByColumns` parameter (the default) converts a ColdFusion query into a row-oriented JSON Object with the following elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMNS</td>
<td>An array of the names of the columns.</td>
</tr>
<tr>
<td>DATA</td>
<td>A two-dimensional array, where:</td>
</tr>
<tr>
<td></td>
<td>• Each entry in the outer array corresponds to a row of query data.</td>
</tr>
<tr>
<td></td>
<td>• Each entry in the inner arrays is a column field value in the row, in the same order as the COLUMNS array entries.</td>
</tr>
</tbody>
</table>

For example, the `SerializeJSON` function with a `serializeQueryByColumns` parameter value of `false` converts a ColdFusion query with two columns, City, and State, and two rows of data into following format:

```
{ "COLUMNS": ["CITY", "STATE"], "DATA": [["Newton", "MA"], ["San Jose", "CA"]]
```

The `SerializeJSON` function with a `true serializeQueryByColumns` parameter value of `true` converts a ColdFusion query into a column-oriented JSON Object that is equivalent to the WDDX query representation. The JSON Object has three elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROWCOUNT</td>
<td>The number of rows in the query.</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>An array of the names of the columns.</td>
</tr>
<tr>
<td>DATA</td>
<td>An Object with the following:</td>
</tr>
<tr>
<td></td>
<td>• The keys are the query column names</td>
</tr>
<tr>
<td></td>
<td>• The values are arrays that contain the column data</td>
</tr>
</tbody>
</table>

The `SerializeJSON` function with a `true serializeQueryByColumns` parameter value of `true` converts a ColdFusion query with two columns, City, and State, and two rows of data into following format:
Note

The `SerializeJSON` function generates an error if you try to convert binary data into JSON format.

The `SerializeJSON` function converts all other ColdFusion data types to the corresponding JSON types. It converts structures to JSON Objects, arrays to JSON Arrays, numbers to JSON Numbers, and strings to JSON Strings.

Note

ColdFusion internally represents structure key names using all-uppercase characters, and, therefore, serializes the key names to all-uppercase JSON representations. Any JavaScript that handles JSON representations of ColdFusion structures must use all-uppercase structure key names, such as `CITY` or `STATE`. You also use the all-uppercase names `COLUMNS` and `DATA` as the keys for the two arrays that represent ColdFusion queries in JSON format.

Example

This example creates a JSON-format data feed with simple weather data for two cities. The data feed is in the form of a JavaScript application that consists of a single function call that has a JSON Object as its parameter. The example code does the following:

1. Creates a query object with two rows of weather data. Each row has a city, the current temperature, and an array of forecast structures, with each with the high, low, and weather prediction for one day. Normally, datasource provides the data; to keep the example simple, the example uses the same prediction for all cites and days.
2. Converts the query to a JSON format string and surrounds it in a JavaScript function call.

Writes the result to the output.

If you view this page in your browser, you see the resulting JavaScript function and JSON parameter. To use the results of this page in an application, put this file and the example for the `DeserializeJSON` function in an appropriate location under your ColdFusion web root, replace the URL in the `DeserializeJSON` example code with the correct URL for this page, and run the `DeserializeJSON` example.
ColdFusion 11 has enhanced the JSON serialization to support the following new features:

1. Case preservation of struct keys
2. Data type preservation
3. Key-value serialization of CF Query
4. Custom serializers

**Case preservation of struct keys**

Currently, the cases for struct keys are not preserved in ColdFusion. The struct keys get converted to upper case automatically.

For instance, consider the following code:
<cfscript>
    data = {empName="James", age="26"};
    serializedStr = serializeJSON(data);
    writeoutput(serializedStr);
</cfscript>

In ColdFusion 10 and earlier versions, the output generated by the above code will be:

{ 'EMPNAME'='', 'AGE'='' }

For the release after ColdFusion 11, the output generated will be:

{ 'empName'='', 'age'='' }

To enable case preservation of struct keys at the application level, modify the application.cfc file by setting:

```
this.serialization.preservecaseforstructkey = true
```

The default behavior is true. To switch back to the old behavior, set this value to false.

To enable case preservation of struct keys at the server level, perform the following tasks:

1. From the ColdFusion Administrator page, click Server Settings > Settings
2. Click Preserve case for Struct keys for Serialization

Note that this setting is used during compilation of the CFML page and therefore if this flag is changed (in the administrator or programatically), any pages relying on the change must be recompiled. This is done typically by simply editing the file (make any change at all) and re-executing it. If “trusted cache” is enabled in the ColdFusion Administrator, you must clear the template cache (of at least those affected files), which can also be done from within the ColdFusion Administrator Caching page.

**Data type preservation**

The ColdFusion language is typeless and does not evaluate or preserve the type information at the time of code generation. At runtime, ColdFusion tries to make its best guess to determine the datatype, which may cause some unexpected behavior. For example, at the time of JSON serialization, ColdFusion attempts at converting a string to a
number. If the attempt is successful, then the passed data type is treated as number irrespective of whether you wanted it to be a string or not.

Starting from ColdFusion 11, the data type is preserved during the code execution time for Query and CFCs.

SerializeJSON considers datatypes defined in the database for serialization. If the database defines a column as a string, any number inserted into the column will still be treated as a string by SerializeJSON.

For instance,

```cfc
<cfquery name="qry_Users" datasource="#DSN#">
    select * from users
</cfquery>
<cfoutput>#SerializeJSON(qry_Users)#</cfoutput><br>
```

Serializing query

SerializeJSON honors the datatypes of columns defined in the database. The same would work for in-memory queries created using QueryNew() as long as the datatype is specified for the columns.

Consider the CFC property type example:

`Employee.cfc`

```cfc
Component accessors="true"
{
    property string empName;
    property numeric age;
    property string dept;
}
```

`Index.cfm`

```cft
<cfscript>
    emp = new Employee({empName="James", age=26, dept="000"});
    writeOutput(SerializeJSON(emp));
</cfscript>
```

OUTPUT: {"dept":"000","empName":"James","age":26}

In the previous version of ColdFusion, 000 will be automatically converted to a number at runtime.
Additional format for query serialization

ColdFusion 10 supports 2 different ways to serialize a query object to a JSON string:

- Using row
- Using column

However, these 2 types are not the easiest to use with AJAX applications. ColdFusion 11 introduces a new way to serialize a query object to a JSON string:

- **Using struct**

All the 3 ways, can be defined at the application-level and will be used in serialized JSON functions if the type is not defined explicitly. In application.cfc, define:

```coldfusion
this.serialization.serializeQueryAs = [row|column|struct]
```

Note that "struct" is also available to be accessible through an AJAX argument. Now you can pass struct in an AJAX URL to serialize a query object as struct.

ColdFusion 11 now supports serializing the query object to a JSON string that is AJAX-friendly:

```javascript
[
  {"colour":"red","id":1},
  {"colour":"green","id":2},
  {"colour":"blue","id":3}
]
```

The current `SerializeJSON` function has been enhanced to support the 'key-value' format.

```coldfusion
SerializeJSON( Object o, Object serializeQueryByColumns, boolean secure, boolean useCustomSerializer);
```

If you are using the serializeQueryAs property in application.cfc, you do not need to specify the serializeQueryByColumns property unless you need to override the functionality.

**Custom serializers**

In the application.cfc file, you can register your own handler for serializing and deserializing the complex types. If the serializer is not specified, ColdFusion uses the default mechanism for serialization. For more information see [Support for pluggable serializer and deserializer](#).
SessionInvalidate

Description

Invalidates or cleans up the current session.

⚠️ Note

The `sessionInvalidate()` method does not invalidate the underlying J2EE session.

Returns

None

Category

Display and formatting functions

Syntax

`sessionInvalidate()`

See also

`SessionRotate`

History

ColdFusion 10: Added this function.

Parameters

None

Usage

Use this function to invalidate the existing session.

Example

Application.cfc

```
<cfcomponent>
  <cfset this.sessionManagement = true />
  <cfset this.name = "session_app" />
</cfcomponent>
```

sessionInvalidate.cfm
<cfif isDefined("url.invalidate")>
  <cfset sessionInvalidate()/>
</cfif>
<cfif isDefined("url.name")>
  <cfset session.name = url.name />
</cfif>
<cfdump var="#session#" label="SESSION">
<cfoutput>
<a href="sessionInvalidate.cfm?name=BOB">Set session.name = BOB</a> <br/>
<a href="sessionInvalidate.cfm?invalidate=TRUE">Invalidate the session</a>
</cfoutput>
SessionRotate

Description

Renews the session when started. For example, you want to generate a new session after a successful login. It prevents session attacks, because the session before and after a successful authentication is different. The method, 

- Creates a session
- Copies the data from the old session to the new session
- Invalidates the old session
- Invalidates or overwrites the old session cookies
- Creates new session cookies if the old session cookies are invalidated
- Copies and updates client storage data to new session keys

Returns

None

Category

Display and formatting functions

Syntax

SessionRotate()

See also

SessionInvalidate

History

ColdFusion 10: Added this function.

Parameters

None

Usage

Use this function to rotate the session.

Example

Application.cfc

```cfc
<cfcomponent>
<cfset this.sessionManagement = true />
<cfset this.name = "session_app" />
</cfcomponent>
```

sessionRotate.cfm
<cfif isDefined("url.rotate") >
  <cfset sessionRotate()/>
</cfif>
<cfif isDefined("url.name") >
  <cfset session.name = url.name />
</cfif>
<cfdump var="#session#" label="SESSION">
  <cfoutput>
    <a href="sessionRotate.cfm?name=BOB">Set session.name = BOB </a> <br/>
    <a href="sessionRotate.cfm?rotate=TRUE">Rotate the session</a>
  </cfoutput>
</cfoutput>
SessionGetMetaData

Description

Returns meta data related to the current session.

Returns

Meta data structure related to the current session

<table>
<thead>
<tr>
<th>Structure element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>starttime</td>
<td>The start time of the current session.</td>
</tr>
</tbody>
</table>

Syntax

sessionGetMetaData()

History

ColdFusion 10: Added this function.

Example

```cf
<cfset metaData = sessionGetMetaData() >
```
SetEncoding

Description

Sets the character encoding (character set) of Form and URL scope variable values; used when the character encoding of the input to a form, or the character encoding of a URL, is not in UTF-8 encoding.

Returns

None

Category

International functions, System functions

Function syntax

SetEncoding(scope_name, charset)

See also

GetEncoding, cfcontent, cfprocessingdirective, URLDecode, URLEncodedFormat; Locales in the Developing ColdFusion Applications

History

ColdFusion MX: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| scope_name | • url  
| | • form  
| charset | The character encoding in which text in the scope variables is encoded. The following list includes commonly used values:  
| | • utf-8  
| | • iso-8859-1  
| | • windows-1252  
| | • us-ascii  
| | • shift_jis  
| | • iso-2022-jp  
| | • euc-jp  
| | • euc-kr  
| | • big5  
| | • euc-cn  
| | • utf-16  

Usage
Use this function when the character encoding of the input to a form or the character encoding of a URL is not in UTF-8 encoding. For example, Traditional Chinese characters are often in Big5 encoding. This function resets URL and Form variables, so you should call it before using these variables (typically, in the Application.cfm page or Application.cfc file). Calling this function first also avoids interpreting the characters of the variables incorrectly. For more information on character encoding, see the following web pages:

- [www.w3.org/International/O-charset.html](http://www.w3.org/International/O-charset.html) provides general information on character encoding and the web, and has several useful links.
- [www.iana.org/assignments/character-sets](http://www.iana.org/assignments/character-sets) is a complete list of character sets names used on the Internet, maintained by the Internet Assigned Numbers Authority.
- [java.sun.com/j2se/1.4.1/docs/guide/intl/encoding.doc.html](http://java.sun.com/j2se/1.4.1/docs/guide/intl/encoding.doc.html) lists the character encoding that Java 1.4.1, and therefore the default ColdFusion configuration, can interpret. If you use a JVM that does not conform to the Sun Java 2 Platform, Standard Edition, v 1.4.1, the supported locales may differ. The list uses Java internal names, not the IANA character encoding names that you normally use in the SetEncoding charset parameter and other ColdFusion attributes and parameters. Java automatically converts standard IANA names to its internal names as needed.

Example

```cfc
<!--- This example sends and interprets the contents of two fields as big5 encoded text. Note that the form fields are received as URL variables because the form uses the GET method. --->
<cfcontent type="text/html; charset=big5">
<form action="#cgi.script_name#" method='get'>
<input name='xxx' type='text'>
<input name='yyy' type='text'>
<input type="Submit" value="Submit">
</form>
<cfif IsDefined("URL.xxx")>
<cfscript>
SetEncoding("url", "big5");
WriteOutput("URL.XXX is " & URL.xxx & "<br>");
WriteOutput("URL.YYY is " & URL.yyy & "<br>");
theEncoding = GetEncoding("URL");
WriteOutput("The URL variables were decoded using " & theEncoding & ": encoding.");
WriteOutput("The encoding is " & theEncoding);
</cfscript>
</cfif>
```
SetLocale

Description

Sets the country/language locale for ColdFusion processing and the page returned to the client. The locale value determines the default format of date, time, number, and currency values, according to language and regional conventions.

Returns

The locale value prior to setting the new locale, as a string.

Category

International functions, System functions

Function syntax

```
SetLocale(new_locale)
```

See also

GetHttpTimeString, GetLocale, GetLocaleDisplayName; Locales in the Developing ColdFusion Applications

History

ColdFusion MX 7: Added support for all locales supported by the ColdFusion Java runtime.
ColdFusion MX:

- Changed formatting behavior: this function might return a different value than in earlier releases. This function uses Java standard locale determination and formatting rules on all platforms.
- Deprecated the Spanish (Mexican) locale option. It might not work, and it might cause an error, in later releases.
- Changed the Spanish (Modern) option: it now sets the locale to Spanish (Standard).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_locale</td>
<td>The name of a locale; for example, “English (US)”</td>
</tr>
</tbody>
</table>

Usage

You can specify any locale name that is listed in the Server.Coldfusion.SupportedLocales variable. This variable is a comma-delimited list of all locale names supported by the JVM, plus the locale names that were required by ColdFusion prior to ColdFusion MX 7. The following locale names were used in ColdFusion releases through ColdFusion MX 6.1, and continue to be supported. If you use any of these values in the SetLocale function, the GetLocale function returns the name you set, not the corresponding Java locale name.

<table>
<thead>
<tr>
<th>Chinese (China)</th>
<th>French (Belgian)</th>
<th>Korean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese (Hong Kong)</td>
<td>French (Canadian)</td>
<td>Norwegian (Bokmal)</td>
</tr>
</tbody>
</table>
ColdFusion determines the locale value as follows:

- By default, ColdFusion uses the JVM locale, and the default JVM locale is the operating system locale. You can set JVM locale value explicitly in ColdFusion in the ColdFusion Administrator Java and JVM Settings page JVM Arguments field; for example:
  
  ```
  -Duser.language=de -Duser.region=DE.
  ```

- A locale set using the `SetLocale` function persists for the current request or until it is reset by another `SetLocale` function in the request.
- If a request has multiple `SetLocale` functions, the current locale setting affects how locale-sensitive ColdFusion tags and functions, such as the functions that start with LS format data. The last `SetLocale` function that ColdFusion processes before sending a response to the requestor (typically the client browser) determines the value of the response Content-Language HTTP header. The browser that requested the page displays the response according to the rules for the language specified by the Content-Language header.
- ColdFusion ignores any `SetLocale` functions that follow a `cfflush` tag. Because this function returns the previous locale setting, you can save the original locale value. You can restore the original locale by calling `SetLocale` again with the saved variable. For example, the following line saves the original locale in a Session variable:

  ```
  <cfset Session.oldlocale = SetLocale(newLocale)>
  ```

  The variable `server.ColdFusion.SupportedLocales` is initialized at startup with a comma-delimited list of the locales that ColdFusion and the operating system support. If you call `SetLocale` with a locale that is not in the list, the call generates an error.
Note

ColdFusion uses the Spanish (Standard) formats for Spanish (Modern) and Spanish (Standard).

Example

```html
<h3>SetLocale Example</h3>
<p>SetLocale sets the locale to the specified new locale for the current session. A locale encapsulates the set of attributes that govern the display and formatting of date, time, number, and currency values.</p>
<p>The locale for this system is <cfoutput>#GetLocale#</cfoutput>
<p>The old locale was <cfoutput><I>#SetLocale("English (UK)")#</cfoutput>
The locale is now <cfoutput>#GetLocale#</cfoutput>
```
SetProfileString

Description
Sets the value of a profile entry in an initialization file.

Returns
An empty string, upon successful execution; otherwise, an error message.

History
ColdFusion 11: Added the encoding attribute.

Category
System functions

Function syntax

SetProfileString(iniPath, section, entry, value, encoding)

See also
GetProfileSections, GetProfileString

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iniPath</td>
<td>Absolute path of initialization file</td>
</tr>
<tr>
<td>section</td>
<td>Section of the initialization file in which the entry is to be set</td>
</tr>
<tr>
<td>entry</td>
<td>Name of the entry to set</td>
</tr>
<tr>
<td>value</td>
<td>Value to which to set the entry</td>
</tr>
<tr>
<td>encoding</td>
<td>Encoding of the initialization (ini) file. For instance &quot;UTF-8&quot;</td>
</tr>
</tbody>
</table>

Example

```
<h3>SetProfileString Example</h3>
This example uses SetProfileString to set the time-out value in an initialization file. Enter the full path of your initialization file, specify the time-out value, and submit the form.
<cfif Isdefined("Form.Submit")>
```
<cfset IniPath = FORM.iniPath>
<cfset Section = "boot loader">
<cfset MyTimeout = FORM.MyTimeout>
<cfset timeout = GetProfileString(IniPath, Section, "timeout")>

<cfif timeout Is Not MyTimeout>
  <cfif MyTimeout Greater Than 0>
    <hr size = "2" color = ">
    <p>Setting the time-out value to <cfoutput>#MyTimeout#</cfoutput></p>
    <cfset code = SetProfileString(IniPath, Section, "timeout", MyTimeout)>
    <p>Value returned from SetProfileString: <cfoutput>#code#</cfoutput></p>
  </cfif>
  <cfelse>
    <hr size = "2" color = "red">
    <p>The time-out value should be greater than zero in order to provide time for user response.</p>
    <hr size = "2" color = "red">
  </cfelse>
</cfif>

<cfelse>
  <p>The time-out value in your initialization file is already <cfoutput>#MyTimeout#</cfoutput>.</p>
</cfif>

<cfset timeout = GetProfileString(IniPath, Section, "timeout")>
<cfset default = GetProfileString(IniPath, Section, "default")>

<h4>Boot Loader</h4>
<p>The time-out is set to: <cfoutput>#timeout#</cfoutput>.</p>
<p>Default directory is: <cfoutput>#default#</cfoutput>.</p>

</cfif>

<form action = "setprofilestring.cfm">
  <hr size = "2" color = ">
  <table cellspacing = "2" cellpadding = "2" border = "0">
    <tr>
      <td>Full Path of Init File</td>
      <td><input type = "Text" name = "IniPath" value = "C:\myboot.ini"></td>
    </tr>
    <tr>
      <td>Time-out</td>
      <td><input type = "Text" name = "MyTimeout" value = "30"></td>
    </tr>
    <tr>
      <td><input type = "Submit" name = "Submit" value = "Submit"></td>
    </tr>
  </table>
</form>
</table>
</form>
SetVariable

Description

Sets a variable in the `name` parameter to the value of the `value` parameter.

Returns

The new value of the variable.

Category

Dynamic evaluation functions

Function syntax

```
SetVariable(name, value)
```

See also

DE, Evaluate, IIf

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Variable name</td>
</tr>
<tr>
<td>value</td>
<td>A string, the name of a string, or a number</td>
</tr>
</tbody>
</table>

Usage

You can use direct assignment statements in place of this function to set values of dynamically named variables. To do so, put the dynamically named variable in quotation marks and number signs (#); for example:

```
<cfset DynamicVar2 = "ABD">
<cfset "#DynamicVar2#" = "Test Value2">
```

Also, the following lines are equivalent:

```
<cfset "myVar#i#" = myVal>
SetVariable("myVar" & i, myVal)
```

For more information, see Using Expressions and Number Signs in the Developing ColdFusion Applications.

Example
<h3>SetVariable Example</h3>

<cfif IsDefined("FORM.myVariable")>
<!--- strip out url, client., cgi., session., caller. --->
<!--- This example only lets you set form variables --->
<cfset myName = ReplaceList(FORM.myVariable,
    "url,client,cgi,session,caller", "FORM,FORM,FORM,FORM,FORM")>

<cfset temp = SetVariable(myName, FORM.myValue)>
<cfset varName = myName>
<cfset varNameValue = Evaluate(myName)>
<cfoutput>
  <p>Your variable, #varName#</p>
  <p>The value of #varName# is #varNameValue#</p>
</cfoutput>
</cfif>
**Sgn**

**Description**

Determines the sign of a number.

**Returns**

- 1, if *number* is positive.
- 0, if *number* is 0.
- -1, if *number* is negative.

**Category**

[Mathematical functions](#)

**Function syntax**

```
Sgn(number)
```

**See also**

[Abs](#)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A number</td>
</tr>
</tbody>
</table>

**Example**

```
<h3>Sgn Example</h3>
<p>Sgn determines the sign of a number. Returns 1 if number is positive; 0 if number is 0; -1 if number is negative.</p>
<p>Sgn(14): <cfoutput>#Sgn(14)#</cfoutput></p>
<p>Sgn(21-21): <cfoutput>#Sgn(21-21)#</cfoutput></p>
<p>Sgn(-0.007): <cfoutput>#Sgn(-0.007)#</cfoutput></p>
```
Sin

Description
Calculates the sine of an angle that is entered in radians.

Returns
A number; the sine of the angle.

Category
Mathematical functions

Function syntax

```
Sin(number)
```

See also
ASin, Cos, ACos, Tan, Atn, Pi

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Angle, in radians for which to calculate the sine.</td>
</tr>
</tbody>
</table>

Usage
The range of the result is -1 to 1. To convert degrees to radians, multiply degrees by π/180. To convert radians to degrees, multiply radians by 180/π.

⚠️ Note
Because the function uses floating point arithmetic, it returns a very small number (such as 6.1232399574E-017) for angles that should produce 0. To test for a 0 value, check whether the value is less than 0.0000000000001.

Example
<h3>Sin Example</h3>

<!--- Calculate sine if form has been submitted --->
<cfif IsDefined("FORM.sinNum")>

<!--- Make sure input is a number --->
<cfif IsNumeric(#FORM.sinNum#)>

<!--- Convert degrees to radians, call the Sin function. --->
<cfset sinValue=#Sin((Form.sinNum * PI()) / 180)#>

<!--- 0.0000000000001 is the function's precision limit. If absolute value of returned sine value is less, set result to 0 --->
<cfif Abs(sinValue) LT 0.0000000000001>
<cfset sinValue=0>
</cfif>

<!--- Output sine value --->
<cfoutput>
Sin(#FORM.sinNum#) = #sinValue#<br>
</cfoutput>

<!--- If input is not a number, show an error message --->
</cfif>
</cfif>

<form action = "#CGI.script_name#" method="post">
Enter an angle in degrees to get its sine:
<input type = "Text" name = "sinNum" size = "15">
<input type = "Submit" name = "">&nbsp;&nbsp;
<input type = "RESET">
</form>
Sleep

Description
Causes the current thread to stop processing for a specified period of time.

Returns
Does not return a value.

Category
System functions

Syntax

```
Sleep(duration)
```

See also
cfthread, Using ColdFusion Threads in the Developing ColdFusion Applications

History
ColdFusion 8: Added function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>Time, in milliseconds, to stop processing the thread.</td>
</tr>
</tbody>
</table>

Description
The `Sleep` function is useful when one thread must wait until another thread performs some action. The thread that must wait uses the `Sleep` function to stop processing for a time, and, when it awakens, checks to see if the other thread is ready. If it is not, the thread can sleep again. This type of action is useful, for example, when one thread must wait for another thread to complete initialization operations that apply to both threads. The `Sleep` function behaves identically to the `cfthread` tag with an `action` attribute value of `sleep`.

Example
The following example has two threads. The second thread (threadB) uses the `sleep` function to ensure that the first thread (threadA) has completed before it starts processing.
<cfthread name="threadA" action="run">
  <cfset thread.j=1>
  <cfloop index="i" from="1" to="99999">
    <cfset thread.j=thread.j+1>
  </cfloop>
</cfthread>

<cfthread name="threadB" action="run">
<cfscript>
  thread.sleepTimes=0;
  thread.initialized=false;
  while ((threadA.Status != "COMPLETED") && (threadA.Status != "TERMINATED")) {
    sleep(500);
    thread.sleeptimes++;
  }
  // Only do the post-initialization code if the threadA completed.
  if (threadA.Status == "COMPLETED") {
    thread.initialized=true;
    // Post-initialization code would go here.
  }
</cfscript>
</cfthread>

<!Join the threads. --->
<cfthread action="join" name="threadA,threadB" timeout="10000"/>

<!Display the thread information. --->
<!Different actions might be taken based on the thread status information. --->
<cfoutput>
  threadA index value: #threadA.j#<br />
  threadA status: #threadA.Status#<br />
  threadB status: #threadB.Status#<br />
  threadB sleepTimes: #threadB.sleepTimes#<br />
  threadB initialized: #threadB.initialized#<br />
</cfoutput>
SpanExcluding

Description

Gets characters from a string, from the beginning and stops when it encounters any of the characters in a specified set of characters. The search is case sensitive. For example, SpanExcluding("MyString", "inS") excludes "String" from "MyString" and returns "My". Because, in the string "MyString", after "My", the character ‘S’ (which is present in the second string "inS") is encountered.

Returns

A string; characters from string, from the beginning to a character that is in set.

Category

String functions

Function syntax

SpanExcluding(string, set)

See also

GetToken, SpanIncluding; Caching parts of ColdFusion pages in Optimizing ColdFusion applications in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
<tr>
<td>set</td>
<td>A string or a variable that contains one. Must contain one or more characters</td>
</tr>
</tbody>
</table>

Example
<h3>SpanExcluding Example</h3>

<cfif IsDefined("FORM.myString")>
<p>Your string was <cfoutput>#FORM.myString#</cfoutput>
</p>
<p>Your set of characters was <cfoutput>#FORM.mySet#</cfoutput>
</p>
<p>Your string up until one of the characters in the set is:
<cfoutput>#SpanExcluding(FORM.myString, FORM.mySet)#</cfoutput></cfif>

<p>Returns all characters from string from beginning to a character from the set of characters. The search is case-sensitive.

<form method = post action = "spanexcluding.cfm">
  <p>Enter a string:
  <br><input type = "Text" name = "myString" value = "Hey, you!">
  <p>And a set of characters:
  <br><input type = "Text" name = "mySet" value = "Ey">
  <br><input type = "Submit" name = "">
</form>
SpanIncluding

Description

Gets characters from a string from the beginning and stops when it encounters any character that is not in a specified set of characters. The search is case sensitive. For example, SpanIncluding("mystring", "mystery") returns "mystr". Because, in the string "mystring", after "mystr", the character 'i' (which is not present in the second string "mystery") is encountered.

Returns

A string; characters from string, from the beginning to a character that is not in set.

Category

String functions

Function syntax

SpanIncluding(string, set)

See also

GetToken, SpanExcluding; Caching parts of ColdFusion pages in Optimizing ColdFusion applications in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains the search string.</td>
</tr>
<tr>
<td>set</td>
<td>A string or a variable that contains a set of characters. Must contain one or more characters.</td>
</tr>
</tbody>
</table>

Example
<h3>SpanIncluding Example</h3>
<cfif IsDefined("FORM.myString")>
<p>Your string was <cfoutput>#FORM.myString#</cfoutput>
</cfif>
<p>Your set of characters was <cfoutput>#FORM.mySet#</cfoutput></p>
<p>Your string, until the characters in the set have been found, is:
<cfoutput>#SpanIncluding(FORM.myString, FORM.mySet)#</cfoutput></p>
</cfif>
<p>Returns characters of a string, from beginning to a character that is not in set. The search is case-sensitive.
</p>
<form action = "spanincluding.cfm" method="post">
<p>Enter a string:
<br><input type = "Text" name = "myString" value = "Hey, you!">
</p><p>And a set of characters:
<br><input type = "Text" name = "mySet" value = "ey,H">
<br><input type = "Submit" name = "">
</form>
SpreadsheetAddColumn

Description

Adds a column or column data to an Excel spreadsheet object.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

SpreadsheetAddColumn(SpreadsheetObj, data[, startRow, startColumn, insert], datatype);

See also

SpreadsheetAddImage, SpreadsheetAddRow, SpreadsheetAddRows, SpreadsheetDeleteColumn, SpreadsheetDeleteColumns, SpreadsheetFormatColumn, SpreadsheetFormatColumns, SpreadsheetShiftColumns

History

ColdFusion 11: Added the datatype attribute.

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object to which to add the column.</td>
</tr>
<tr>
<td>data</td>
<td>A comma delimited list of cell entries, one per row being added to the column.</td>
</tr>
<tr>
<td>startRow</td>
<td>This parameter is optional. The number of the row at which to start adding the column data. If insert=&quot;true&quot;, all rows in the column above the start row have empty cells. If you omit this parameter the columns are inserted starting at the first row, following the last current column, and you cannot specify a column.</td>
</tr>
<tr>
<td>startColumn</td>
<td>This parameter is optional. The number of the column in which to add the column data.</td>
</tr>
</tbody>
</table>
insert  
This parameter is optional. A Boolean value specifying whether to insert a column. If false, the function replaces data in the specified column entries.

datatype  
The data type denoted by the column. The allowed values are NUMERIC, DATE, and STRING.

Usage

The `spreadsheetAddColumn` function can accept either two or five arguments. You can specify the `spreadsheetAddColumn` function using two parameters as follows:

```cfset spreadsheetAddColumn(SpreadsheetObj, "newcol1,newcol2,newcol3")>```

You can specify the `spreadsheetAddColumn` function using five parameters as follows:

```cfset spreadsheetAddColumn(SpreadsheetObj, "newcol1,newcol2,newcol3", 2, 3, false)>
```

Example

The following example creates an Excel spreadsheet object from a query and inserts a new column 2, with data starting at row 3. The existing columns 2 and greater increment by one.

``` <!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, COURSE_ID, CORNAME
    FROM COURSELIST
</cfquery>

<cfscript>
    ///We need an absolute path, so get the current directory path. 
    theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & courses.xls";
    //Create a new Spreadsheet object and add the query data.
    theSheet = SpreadsheetNew("CourseData");
    SpreadsheetAddRows(theSheet, courses);
    //Insert a new second column to the sheet, with data starting in row 3.
    SpreadsheetAddColumn(theSheet,
        3,2,true);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet" sheet=1
    sheetname="courses" overwrite=true>```
SpreadsheetAddImage

Description

Adds an image to an Excel spreadsheet object.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetAddImage(SpreadsheetObj, imageFilePath, anchor)
SpreadsheetAddImage(SpreadsheetObj, imageData, imageType, anchor)
```

See also

SpreadsheetAddColumn, SpreadsheetAddRow, SpreadsheetAddRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object to which to add the column.</td>
</tr>
<tr>
<td>anchor</td>
<td>The image location, as a comma delimited list in either of the following formats:</td>
</tr>
<tr>
<td></td>
<td>• startRow, startColumn, endRow, endColumn</td>
</tr>
<tr>
<td></td>
<td>• startXPosition, startYPosition, endXPosition, endYPosition, startRow, startColumn, endRow, endColumn</td>
</tr>
<tr>
<td></td>
<td>The first format specifies only the row and column numbers, the second also specifies the positions in the cell, using pixel X and Y coordinates relative to the cell upper left corner. If you use the first format, the image corner positions within the top left and bottom right cell are 0,0 and 0,255.</td>
</tr>
<tr>
<td>imageData</td>
<td>A ColdFusion image object.</td>
</tr>
<tr>
<td>imageFilePath</td>
<td>The absolute path to the image file.</td>
</tr>
</tbody>
</table>
imageType

The image format, one of the following:
- jpg or jpeg
- png
- dib

Usage

Example

The following example creates a PNG format chart, puts it in a new spread sheet as rows 5-12 and column 5-10, and saves the sheet to disk.

```cfml
<cfchart format="png"
    scalefrom="-100" scaleTo="100"
    gridlines="5"
    name="test">
    <cfchartseries type="line">
        <cfchartdata item="Point1" value="-50">
        <cfchartdata item="Point2" value="-25">
        <cfchartdata item="Point3" value="1">
        <cfchartdata item="Point4" value="25">
        <cfchartdata item="Point5" value="50">
        <cfchartdata item="Point6" value="75">
        <cfchartdata item="Point7" value="99">
    </cfchartseries>
</cfchart>

<cfscript>
    theDir=GetDirectoryFromPath(GetCurrentTemplatePath());
    SpreadsheetObj=SpreadsheetNew();
    SpreadsheetAddImage(SpreadsheetObj,test,"png","5,5,12,10");
</cfscript>

<cfspreadsheet action="write" name=SpreadsheetObj filename="#theDir#imagesheet.xls"
    sheetname="chart" overwrite=true>
SpreadsheetAddFreezePane

Description

Locks or freezes specific rows or columns in the worksheet.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetAddFreezePane(spreadsheetobj, freezcol, freezrow[, col, row])
```

See also

SpreadsheetAddColumn, SpreadsheetAddImage, SpreadsheetAddRow, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobj</td>
<td>The Excel spreadsheet object to which to add the freeze pane.</td>
</tr>
<tr>
<td>freezcol</td>
<td>Specifies the column boundary of the freeze pane. The columns contained within the column boundary are frozen, while the rest of the worksheet scrolls.</td>
</tr>
<tr>
<td>freezrow</td>
<td>Specifies the row boundary of the freeze pane. The rows contained within the row boundary are frozen, while the rest of the worksheet scrolls.</td>
</tr>
<tr>
<td>col</td>
<td>This parameter is optional. The column that should appear next to the freezcol that you specify. This parameter is useful in hiding data. For example, in a worksheet, you can specify column 5 to appear immediately after column 2, and hide column 3 and column 4.</td>
</tr>
</tbody>
</table>
### row

This parameter is optional. The row that should appear next to the `freezrow` that you specify. This parameter is useful in hiding data. For example, in a worksheet, you can specify row 10 to appear immediately after row 7, and hide row 8 and row 9.

#### Usage

You can keep an area of the worksheet visible while scrolling to another area of the worksheet by freezing panes. When you freeze a pane, you lock or freeze specific rows and columns in the worksheet. The rows and columns that are frozen in the worksheet are indicated by a solid line.

#### Note

You cannot split the pane into two worksheet areas.

#### Example

The following example freezes the spreadsheet at column 3 and row 2 of the worksheet.

```javascript
SpreadSheetAddFreezePane(SpreadsheetObj, 3, 2);
```

The following example freezes the spreadsheet at column 3 and row 2, and hides data in column 4 and rows 3 to 10.

```javascript
SpreadSheetAddFreezePane(SpreadsheetObj, 3, 2, 5, 10);
```
SpreadsheetAddInfo

Description

Sets document properties for a new spreadsheet or modifies properties for an existing spreadsheet.

Returns

This function does not return a value.

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetAddInfo(spreadsheetobj, property_struct)
```

See also

SpreadsheetAddColumn, SpreadsheetAddImage, SpreadsheetAddRow, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobj</td>
<td>The Excel spreadsheet object from which to get the value.</td>
</tr>
<tr>
<td>property_struct</td>
<td>The following properties of the spreadsheet can be modified or set:</td>
</tr>
<tr>
<td></td>
<td>• AUTHOR</td>
</tr>
<tr>
<td></td>
<td>• CATEGORY</td>
</tr>
<tr>
<td></td>
<td>• LASTAUTHOR</td>
</tr>
<tr>
<td></td>
<td>• COMMENTS</td>
</tr>
<tr>
<td></td>
<td>• KEYWORDS</td>
</tr>
<tr>
<td></td>
<td>• MANAGER</td>
</tr>
<tr>
<td></td>
<td>• COMPANY</td>
</tr>
<tr>
<td></td>
<td>• SUBJECT</td>
</tr>
<tr>
<td></td>
<td>• TITLE</td>
</tr>
</tbody>
</table>

Usage

This function is supported by Microsoft Office Excel 2007 (.xlsx) and Microsoft Office 2003 (.xls).
<cfspreadsheet action="read" src="#filename#" name="a" />
<cfset info = StructNew()>
<cfset info.title="Title">
<cfset info.category="Category test">
<cfset info.author="ABC">
<cfset info.comments="Comments for this file">
<cfset spreadsheetaddInfo(a,info)>
<cfspreadsheet action="write" filename="#dirname#SingleSheet.xls" name=a overwrite="yes"/>
SpreadsheetAddRow

Description

Adds a row to an Excel spreadsheet object.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

SpreadsheetAddRow(spreadsheetObj, data [,row, column, insert, datatype])

See also

SpreadsheetAddColumn, SpreadsheetAddImage, SpreadsheetAddRows, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 11: Added the datatype attribute.

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object to which to add the column.</td>
</tr>
<tr>
<td>data</td>
<td>A comma delimited list of cell entries, one per column.</td>
</tr>
<tr>
<td>row</td>
<td>The number of the row to insert. The row numbers of any existing rows with numbers equal to or greater than this value are incremented by one. If you specify a value for this parameter, you must also specify a value for column. If you omit this parameter the rows are inserted following the last current row, and you cannot specify a column.</td>
</tr>
<tr>
<td>column</td>
<td>The number of the column in which to add the column data. All columns in the row to the left of the start column have empty cells. If you specify a value for this parameter, you must also specify a value for row.</td>
</tr>
</tbody>
</table>
**insert**

This parameter is optional. The default value is `true`. A Boolean value specifying whether to insert a row. If `false`, the function replaces data in the specified row entries.

**datatype**

The data type denoted by the row. The allowed values are `NUMERIC`, `DATE`, and `STRING`.

### Usage

### Example

The following example adds a row of data as row 10. The data starts at column 2, and any existing row numbers 10 and higher increment by one.

```coldfusion
<!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, COURSE_ID, CORNAME
    FROM COURSELIST
</cfquery>

<cfscript>
    /// We need an absolute path, so get the current directory path.
    theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & courses.xls"
    // Create a new Spreadsheet object and add the query data.
    theSheet = SpreadsheetNew("CourseData");

    SpreadsheetAddRows(theSheet, courses);
    // Insert a new eighth row to the sheet, with data starting in column 1.
    SpreadsheetAddRow(theSheet, "150, ENGL, 95, Poetry 1", 8, 1);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheet=1 sheetname="courses" overwrite=true>
```
SpreadsheetAddRows

Description

Adds multiple rows from a query to an Excel spreadsheet object.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

`SpreadsheetAddRows(spreadsheetObj, data[, row, column, insert], datatype)`

See also

`SpreadsheetAddColumn`, `SpreadsheetAddImage`, `SpreadsheetAddRow`, `SpreadsheetDeleteRow`, `SpreadsheetDeleteRows`, `SpreadsheetFormatRow`, `SpreadsheetFormatRows`, `SpreadsheetShiftRows`

History

ColdFusion 11: Added the datatype attribute

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>spreadsheetObj</code></td>
<td>The Excel spreadsheet object to which to add the column.</td>
</tr>
<tr>
<td><code>data</code></td>
<td>A query object with the row data or an array.</td>
</tr>
<tr>
<td><code>row</code></td>
<td>The number of the row at which to insert the rows. The row numbers of any existing rows with numbers equal to or greater than this value are incremented by number of added rows. If you specify a value for this parameter, you must also specify a value for <code>{column}</code>. If you omit this parameter the rows are inserted following the last current row.</td>
</tr>
<tr>
<td><code>column</code></td>
<td>The number of the column in which to add the column data. All columns in the row to the left of the start column have empty cells. If you specify a value for this parameter, you must also specify a value for <code>row</code>.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>insert</td>
<td>This parameter is optional. The default value is <code>true</code>. A Boolean value specifying whether to insert a row. If <code>false</code>, the function replaces data in the specified row entries.</td>
</tr>
<tr>
<td>datatype</td>
<td>The data type denoted by the row. The allowed values are NUMERIC, DATE, and STRING.</td>
</tr>
</tbody>
</table>

**Example**

The following example creates a spreadsheet by creating a new Excel spreadsheet object and using the `AddRows` function to add the data from a query.

```cfml
<!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, COURSE_ID, CORNAME
    FROM COURSELIST
</cfquery>

<cfscript>
/// We need an absolute path, so get the current directory path.
theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & courses.xls"
//Create a new Excel spreadsheet object and add the query data.
theSheet = SpreadsheetNew("CourseData");

SpreadsheetAddRows(theSheet, courses);
SpreadsheetAddRows(theSheet, ["1,a", "2,B,b"]);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheet=1 sheetname="courses" overwrite=true>
```

The following example shows the usage of the `datatype` parameter:

```cfml
<!--- Using the datatype parameter --->
<cfset xlObj = spreadsheetNew("testsheets", true)>
<cfset spreadsheetAddRows(xlObj, val_arr, 1, 1, true,
    ["INTEGER:1;STRING:2,3,4;DATE:"])>
<cfset spreadsheetwrite(xlObj, 
    
    #expandpath("./")#test_datatype.xlsx", 
    "", true)>
```
SpreadsheetAddSplitPane

Description

Splits panes into four separate worksheet areas.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

SpreadsheetAddSplitPane(spreadsheetobj, x-position, y-position, splitcol, splitrow [,position])

See also

SpreadsheetAddColumn, SpreadsheetAddImage, SpreadsheetAddRow, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobj</td>
<td>The Excel spreadsheet object to which to add the split pane.</td>
</tr>
<tr>
<td>x-position</td>
<td>Specifies the x-axis position of the first quadrant. The x-position value is 1/20th the value of a pixel in the worksheet.</td>
</tr>
<tr>
<td>y-position</td>
<td>Specifies the y-axis position of the first quadrant. The y-position value is 1/20th the value of a pixel in the worksheet.</td>
</tr>
<tr>
<td>splitcol</td>
<td>Specifies the columns that appear in quadrant 2 of the spreadsheet.</td>
</tr>
<tr>
<td>splitrow</td>
<td>Specifies the rows that appear in quadrant 3 of the spreadsheet.</td>
</tr>
</tbody>
</table>
position

This attribute is optional. Specifies the position to apply the split bar to split the pane. It can be one of the following:

- LOWER_LEFT
- LOWER_RIGHT
- UPPER_RIGHT
- UPPER_LEFT

Usage

You can split the pane into four worksheet areas in the spread sheet. The split is applied at the pixel level. You can adjust the worksheet area by dragging the split bar, as required.

⚠️ **Note**

You cannot split the pane into two worksheet areas.

Example

The following example splits the spreadsheet into four quadrants. The x and y positions are at 2000 and 2000 values. Column 5 of the spreadsheet appears in quadrant 2; and row 7 of the spreadsheet appears in quadrant 7. The split bar appears at the lower left side of the spreadsheet.

```javascript
SpreadSheetAddSplitPane(spreadsheetobj, 2000, 2000, 5, 7, LOWER_LEFT);
```
SpreadsheetCreateSheet

Description

Creates an additional spreadsheet.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetCreateSheet(spreadsheetObj, [sheetname])
```

See also

SpreadsheetAddColumn, SpreadsheetDeleteColumns, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatColumn, SpreadsheetFormatColumns, SpreadsheetShiftColumns, SpreadsheetSetActiveSheet

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object from which you create the additional sheet.</td>
</tr>
<tr>
<td>sheetname</td>
<td>Name of the new sheet. This is optional.</td>
</tr>
</tbody>
</table>

Example

The following example creates two sheets: CourseData and EvaluationSheet.
<cfquery name="courses" datasource="cfdocexamples"
cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, COURSE_ID, CORNAME
    FROM COURSELIST
</cfquery>

<cfscript>

    theFile = GetDirectoryFromPath(GetCurrentTemplatePath()) & "courses.xls";
    theSheet = SpreadsheetNew("CourseData");

    SpreadsheetAddRows(theSheet, courses);
    SpreadsheetCreateSheet (theSheet, "EvaluationSheet");
</cfscript>

<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
sheet=1 sheetname="courses" overwrite=true>
**SpreadsheetDeleteColumn**

**Description**

Deletes the data from a column of an Excel spreadsheet object. It does not delete the column.

**Returns**

Does not return a value.

**Category**

Microsoft Office Integration

**Function syntax**

```
SpreadsheetDeleteColumn(spreadsheetObj, column)
```

**See also**

`SpreadsheetAddColumn`, `SpreadsheetDeleteColumns`, `SpreadsheetDeleteRow`, `SpreadsheetDeleteRows`, `SpreadsheetFormatColumn`, `SpreadsheetFormatColumns`, `SpreadsheetShiftColumns`

**History**

ColdFusion 9: Added the function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object from which to delete the columns.</td>
</tr>
<tr>
<td>column</td>
<td>The column to delete.</td>
</tr>
</tbody>
</table>

**Example**

The following example deletes the data from column 2 from a spreadsheet.
<!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
        SELECT CORNUMBER, DEPT_ID, COURSE_ID, CORNAME
        FROM COURSELIST
</cfquery>

<cfscript>
    /// We need an absolute path, so get the current directory path.
    theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & courses.xls";
    // Create a new Excel spreadsheet object and add the query data.
    theSheet = SpreadsheetNew("CourseData");
    SpreadsheetAddRows(theSheet,courses);
    // Delete the second column of the sheet.
    SpreadsheetDeleteColumn(theSheet,2);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheet=1 sheetname="courses" overwrite=true>
SpreadsheetDeleteColumns

Description

Deletes the data from multiple columns of an Excel spreadsheet object. This function does not remove the columns.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetDeleteColumns(spreadsheetObj, range)
```

See also

SpreadsheetAddColumn, SpreadsheetDeleteColumn, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatColumn, SpreadsheetFormatColumns, SpreadsheetShiftColumns

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object from which to delete the columns.</td>
</tr>
<tr>
<td>range</td>
<td>A string containing the columns to delete, using any combination of the following formats:</td>
</tr>
<tr>
<td></td>
<td>• <code>startColumn-endColumn</code> - Delete columns in a single range.</td>
</tr>
<tr>
<td></td>
<td>• <code>column,column,column</code>... - Delete one or more individual columns.</td>
</tr>
<tr>
<td></td>
<td>You can also provide both the formats together. For example, 1{{, 2, 3-5, 7-12}}.</td>
</tr>
</tbody>
</table>

Example

The following example deletes the data from columns 2-4 and column 6 from a spreadsheet.
<!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, CORLEVEL, COURSE_ID, CORNAME, CORDESC, LASTUPDATE
    FROM COURSELIST
</cfquery>

<cfscript>
    /// We need an absolute path, so get the current directory path.
    theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & courses.xls";
    // Create a new Excel spreadsheet object and add the query data.
    theSheet = SpreadsheetNew("CourseData");

    SpreadsheetAddRows(theSheet,courses);
    // Delete columns 2 though 4 and 6.
    SpreadsheetDeleteColumns(theSheet,"2-4,6");
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheet=1 sheetname="courses" overwrite=true>
SpreadsheetDeleteRow

Description

Deletes all data from a row of an Excel spreadsheet object. It does not delete the row.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```plaintext
SpreadsheetDeleteRow(spreadsheetObj, row)
```

See also

SpreadsheetDeleteColumn, SpreadsheetDeleteColumns, SpreadsheetDeleteRows, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object from which to delete the row.</td>
</tr>
<tr>
<td>row</td>
<td>The row to delete.</td>
</tr>
</tbody>
</table>

Example

The following example deletes row 10 from a spreadsheet.
<cfquery name="courses" datasource="cfdocexamples" cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
   SELECT CORNUMBER, DEPT_ID, CORLEVEL, COURSE_ID, CORNAME, CORDESC, LASTUPDATE FROM COURSELIST
</cfquery>

<cfscript>
   /// We need an absolute path, so get the current directory path.
   theFile = GetDirectoryFromPath(GetCurrentTemplatePath()) & "courses.xls";
   // Create a new Excel spreadsheet object and add the query data.
   theSheet = SpreadsheetNew("CourseData");
   SpreadsheetAddRows(theSheet, courses);

   // Delete row 10.
   SpreadsheetDeleteRow(theSheet, "10");
</cfscript>

<cfspreadsheet action="write" filename="#theFile#" name="theSheet" sheet=1 sheetname="courses" overwrite=true>
SpreadsheetDeleteRows

Description

Deletes all data from multiple rows of an Excel spreadsheet object. It does not delete the row.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetDeleteRows(spreadsheetObj, range)
```

See also

SpreadsheetDeleteColumn, SpreadsheetDeleteColumns, SpreadsheetDeleteRow, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object in which to delete the rows.</td>
</tr>
<tr>
<td>range</td>
<td>The rows to delete, using any combination of the following form:</td>
</tr>
<tr>
<td></td>
<td>- <code>startRow-endRow</code> - Insert rows in a single range.</td>
</tr>
<tr>
<td></td>
<td>- <code>row,row,row...</code> - Insert one or more individual rows.</td>
</tr>
<tr>
<td></td>
<td>You can also provide both the formats together. For example, 1, 2, 3–5, 7–12.</td>
</tr>
</tbody>
</table>

Example

The following example deletes rows 1 and 5-10 from a spreadsheet.
<!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, CORLEVEL, COURSE_ID, CORNAME, CORDESC, LASTUPDATE
    FROM COURSELIST
</cfquery>

<cfscript>
///We need an absolute path, so get the current directory path.
theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & courses.xls";
//Create a new Excel spreadsheet object and add the query data.
theSheet = SpreadsheetNew("CourseData");

SpreadsheetAddRows(theSheet,courses);
//Delete rows 1 and 5 though 10.
SpreadsheetDeleteRows(theSheet,"1,5-10");
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheet=1 sheetname="courses" overwrite=true>
SpreadsheetFormatCell

Description

Formats the contents of a single cell of an Excel spreadsheet object.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

<table>
<thead>
<tr>
<th>Function syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpreadsheetFormatCell(spreadsheetObj, format, row, column)</td>
</tr>
</tbody>
</table>

See also

SpreadsheetFormatColumn, SpreadsheetFormatColumns, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object to which to set the format.</td>
</tr>
<tr>
<td>format</td>
<td>A structure containing the format information. For details see Usage.</td>
</tr>
<tr>
<td>row</td>
<td>The row number of the cell.</td>
</tr>
<tr>
<td>column</td>
<td>The column number of the cell.</td>
</tr>
</tbody>
</table>

Usage

The format structure can specify any or all of the following values

<table>
<thead>
<tr>
<th>Name</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignment</td>
<td>left (default), right, center, justify, general, fill, and center_selection</td>
</tr>
<tr>
<td>bold</td>
<td>A Boolean value. The default value is false.</td>
</tr>
<tr>
<td>bottomborder</td>
<td>A border format, any of the following:none (default), thin, medium, dashed, hair, thick, double, dotted, medium_dashed, dash_dot, medium_dash_dot, dash_dot_dot, medium_dash_dot_dot, slanted_dash_dot</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bottombordercolor</td>
<td>See the color field for the complete list of colors.</td>
</tr>
<tr>
<td>color</td>
<td>Any value in the org.apache.poi.hssf.util.HSSFColor class: black, brown, olive_green, dark_green, dark_teal, dark_blue, indigo, grey_80_percent, orange, dark_yellow, green, teal, blue, blue_grey, grey_50_percent, red, light_orange, lime, sea_green, aqua, light_blue, violet, grey_40_percent, pink, gold, yellow, bright_green, turquoise, dark_red, sky_blue, plum, grey_25_percent, rose, light_yellow, light_green, light_turquoise, light_turquoise, pale_blue, lavender, white, cornflower_blue, lemon_chiffon, maroon, orchid, coral, royal_blue, light_cornflower_blue</td>
</tr>
<tr>
<td>dataformat</td>
<td>An Excel data format. Most of the formats supported by MS Excel are supported. The following are the built-in formats:</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><img src="dataformat_table.png" alt="Formatted data table" /></td>
</tr>
</tbody>
</table>

<p>| fgcolor | See the color field for the complete list of colors. |
| fillpattern | Any of the following: big_spots (default), squares, nofill, solid_foreground, fine_dots, alt_bars, sparse_dots, thick_horz_bands, thick_vert_bands, thick_backward_diag, thick_forward_diag, diamonds, less_dots, least_dots |
| font | A valid system font name. |</p>
<table>
<thead>
<tr>
<th><strong>fontsize</strong></th>
<th>An integer point value.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hidden</strong></td>
<td>A Boolean value. The default value is false.</td>
</tr>
<tr>
<td><strong>indent</strong></td>
<td>A positive integer number of default character spaces.</td>
</tr>
<tr>
<td><strong>italic</strong></td>
<td>No value required.</td>
</tr>
<tr>
<td><strong>leftborder</strong></td>
<td>A border format. See bottomborder for valid values.</td>
</tr>
<tr>
<td><strong>leftbordercolor</strong></td>
<td>See the color field for the complete list of colors.</td>
</tr>
<tr>
<td><strong>locked</strong></td>
<td>A Boolean value. The default value is false.</td>
</tr>
<tr>
<td><strong>rightborder</strong></td>
<td>A border format. See bottomborder for valid values.</td>
</tr>
<tr>
<td><strong>rightbordercolor</strong></td>
<td>See the color field for the complete list of colors.</td>
</tr>
<tr>
<td><strong>rotation</strong></td>
<td>An integer number of degrees in the range -90 - 90.</td>
</tr>
<tr>
<td><strong>strikeout</strong></td>
<td>No value required.</td>
</tr>
<tr>
<td><strong>textwrap</strong></td>
<td>A Boolean value. The default value is false.</td>
</tr>
<tr>
<td><strong>topborder</strong></td>
<td>A border format. See bottomborder for valid values.</td>
</tr>
<tr>
<td><strong>topbordercolor</strong></td>
<td>See the color field for the complete list of colors.</td>
</tr>
</tbody>
</table>
| **verticalalignment** | Any of the following: vertical_top, vertical_bottom, vertical_center, vertical_justifyFor example,<cfscript>
|                   | SpreadsheetFormatCellRange(theSheet,{verticalalignment="VERTICAL_TOP"},3,4,30,10);</cfscript> |
| **underline**    | A Boolean value. The default value is false.  |

**Enhancements made in ColdFusion 9.0.1**

You can preformat a cell while you use `SpreadSheetformatcell` as shown in the following example:

```cfs```
<cfscript>
  sheet = SpreadSheetNew();
  SpreadsheetFormatCell(sheet,{dataformat="@"},1,1);
  spreadsheetSetCellValue(sheet,'000006534',1,1);
</cfscript>
```

Here, the cell is preformatted and the data is taken as it is provided.
Example

The following example creates a sheet, sets a simple format for the cell at row 3 column 4, and writes the result to a file:

```cfc
<!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, CORLEVEL, COURSE_ID, CORNAME, CORDESC, LASTUPDATE
    FROM COURSELIST
</cfquery>

<cfscript>
    ///We need an absolute path, so get the current directory path.
    theFile = GetDirectoryFromPath(GetCurrentTemplatePath()) & "courses.xls";
    //Create a new Excel spreadsheet object and add the query data.
    theSheet = SpreadsheetNew("CourseData");
    SpreadsheetAddRows(theSheet, courses);
    // Define a format for the cell.
    format1 = StructNew();
    format1.font = "serif";
    format1.fontsize = "12";
    format1.color = "dark_green";
    format1.bold = "true";
    format1.alignment = "center";
    SpreadsheetFormatCell(theSheet, format1, 3, 4);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheet=1 sheetname="courses" overwrite=true>

The following examples show how to use dataformat:
<cfset a = spreadsheetnew()>  
<cfset format = structnew()>  
<cfset format.dataformat = "0.00">  
<cfset spreadsheetaddrow(a,"1,2,3,4",2,1)>  
<cfset spreadsheetformatrow(a,format,2)>  
<cfset format.dataformat = "0.00%">  
<cfset spreadsheetaddrow(a,"1,2,3,4",4,1)>  
<cfset spreadsheetformatrow(a,format,4)>  
<cfset format.dataformat = "0.00E+00">  
<cfset spreadsheetaddrow(a,".00000000000001",5,1)>  
<cfset spreadsheetformatrow(a,format,5)>  
<cfset format.dataformat = "## ??/??">  
<cfset spreadsheetaddrow(a,"3.33",7,1)>  
<cfset spreadsheetformatrow(a,format,7)>  
<cfset format.dataformat = "m/d/yy">  
<cfset spreadsheetaddrow(a,"01/06/09",8,1)>  
<cfset spreadsheetformatrow(a,format,8)>  
<cfset format.dataformat = "##,##0.00">  
<cfset spreadsheetaddrow(a,"2100000",13,1)>  
<cfset spreadsheetformatrow(a,format,13)>  
<cfset format.dataformat = " (##,##0_);(##,##0) ">  
<cfset spreadsheetaddrow(a,"-300",14,1)>  
<cfset spreadsheetformatrow(a,format,14)>  
<cfspreadsheet action="write" filename="#expandpath('.')#/test.xls" name="a" overwrite="true"
SpreadsheetFormatColumn

Description

Formats the contents of a single column of an Excel spreadsheet object.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

SpreadsheetFormatColumn(spreadsheetObj, format, column)

See also

SpreadsheetFormatCell, SpreadsheetFormatColumns, SpreadsheetFormatRow, SpreadsheetFormatRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object in which to set the format.</td>
</tr>
<tr>
<td>format</td>
<td>A structure containing the format information. For details see SpreadsheetFormatCell.</td>
</tr>
<tr>
<td>column</td>
<td>The column number.</td>
</tr>
</tbody>
</table>

Example

The following example creates a sheet, sets a format for column 5, and writes the result to a file:
<!--- Get the spreadsheet data as a query. --->
<cfquery
  name="courses" datasource="cfdocexamples"
  cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
  SELECT CORNUMBER, DEPT_ID, CORLEVEL, COURSE_ID, CORNAME, CORDESC, LASTUPDATE 
  FROM COURSELIST
</cfquery>

<cfscript>
/// We need an absolute path, so get the current directory path.
    theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "courses.xls";
// Create a new Excel spreadsheet object and add the query data.
    theSheet = SpreadsheetNew("CourseData");
    SpreadsheetAddRows(theSheet, courses);
// Define a format for the column.
    format1=StructNew()
    format1.font="Courier";
    format1.fontsize="10";
    format1.color="dark_blue;"
    format1.italic="true";
    format1.bold="true";
    format1.alignment="left";
    format1.textwrap="true";
    format1.fgcolor="pale_blue";
    format1.bottomborder="dotted";
    format1.bottombordercolor="blue_grey";
    format1.leftborder="thick";
    format1.leftbordercolor="blue_grey";
    format1.rightborder="thick";
    format1.rightbordercolor="blue_grey";
    SpreadsheetFormatColumn(theSheet, format1, 5);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="#theSheet#
    sheet=1 sheetname="courses" overwrite=true>
SpreadsheetFormatCellRange

Description

Formats the cells within the given range.

Returns

Nothing

Category

Microsoft Office Integration

Function syntax

SpreadsheetFormatCellRange (spreadsheetObj, format, startRow, startColumn, endRow, endColumn)

See also

SpreadsheetFormatCell, SpreadsheetFormatColumns, SpreadsheetFormatRow, SpreadsheetFormatRows

History

ColdFusion 9.0.1: Added the function. Supports preformatting of a cell while you use this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object for which you want to format the cells.</td>
</tr>
<tr>
<td>format</td>
<td>A structure that contains format information.</td>
</tr>
<tr>
<td>startRow</td>
<td>The number of the first row to format.</td>
</tr>
<tr>
<td>startColumn</td>
<td>The number of the first column to format.</td>
</tr>
<tr>
<td>endRow</td>
<td>The number of the last row to format.</td>
</tr>
<tr>
<td>endColumn</td>
<td>The number of the last column to format.</td>
</tr>
</tbody>
</table>

Example
<!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER,DEPT_ID,CORLEVEL,COURSE_ID,CORNAME,CORDESC,LASTUPDATE
    FROM COURSELIST
</cfquery>

<cfscript>
    ///We need an absolute path, so get the current directory path.
    theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "courses.xls";
    //Create a new Excel spreadsheet object and add the query data.
    theSheet = SpreadsheetNew("CourseData");
    SpreadsheetAddRows(theSheet,courses);
    // Define a format for the column.
    format1=StructNew()
    format1.font="Courier";
    format1.fontsize=10;
    format1.color="dark_blue;";
    format1.italic=true;
    format1.bold=true;
    format1.alignment="left";
    SpreadsheetFormatCellRange(theSheet,format1, 3,4,30,10);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheet=1 sheetname="courses" overwrite=true>
SpreadsheetFormatColumns

Description

Formats the contents of multiple columns of an Excel spreadsheet object.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

SpreadsheetFormatColumns(spreadsheetObj, format, columns)

See also

SpreadsheetFormatCell, SpreadsheetFormatColumn, SpreadsheetFormatRow, SpreadsheetFormatRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object in which to set the format.</td>
</tr>
<tr>
<td>format</td>
<td>A structure containing the format information. For details see SpreadsheetFormatCell.</td>
</tr>
<tr>
<td>columns</td>
<td>The columns to format, in one of the following formats:</td>
</tr>
<tr>
<td></td>
<td>• startColumn-endColumn - Insert columns in a single range.</td>
</tr>
<tr>
<td></td>
<td>• column, column, column... - Insert one or more individual columns.</td>
</tr>
</tbody>
</table>

Example

The following example creates a sheet, sets a format for columns 1-5, and writes the result to a file:
<cfquery>
   name="courses" datasource="cfdocexamples"
   cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
   SELECT CORNUMBER, DEPT_ID, CORLEVEL, COURSE_ID, CORNAME, CORDESC, LASTUPDATE
   FROM COURSELIST
</cfquery>

<cfscript>
/// We need an absolute path, so get the current directory path.
theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "courses.xls";
// Create a new Excel spreadsheet object and add the query data.
theSheet = SpreadsheetNew("CourseData");
SpreadsheetAddRows(theSheet, courses);
// Define a format for the column.
format1=StructNew()
format1.font="Courier"
format1.fontsize="10"
format1.color="dark_blue;"
format1.italic="true"
format1.bold="true"
format1.alignment="left"
format1.textwrap="true"
format1.fgcolor="pale_blue"
format1.bottomborder="dotted"
format1.bottombordercolor="blue_grey"
format1.leftborder="thick"
format1.leftbordercolor="blue_grey"
format1.rightborder="thick"
format1.rightbordercolor="blue_grey"
SpreadsheetFormatColumns(theSheet, format1, "1-5");
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
   sheet=1 sheetname="courses" overwrite=true>
SpreadsheetFormatRow

Description

Formats the contents of a single row of an Excel spreadsheet object.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

SpreadsheetFormatRow(spreadsheetObj, format, row)

See also

Other Spreadsheet* functions([SpreadsheetFormatCell]), SpreadsheetFormatColumn, SpreadsheetFormatColumns, SpreadsheetFormatRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object in which to set the format.</td>
</tr>
<tr>
<td>format</td>
<td>A structure containing the format information. For details see SpreadsheetFormatColumn.</td>
</tr>
<tr>
<td>row</td>
<td>The row number.</td>
</tr>
</tbody>
</table>

Example

The following example creates a sheet, sets a format for rows 1, 3, and 5, and writes the result to a file:
<cfquery>
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
        SELECT CORNUMBER, DEPT_ID, CORLEVEL, COURSE_ID, CORNAME, CORDESC, LASTUPDATE
        FROM COURSELIST
</cfquery>

<cfscript>
    /// We need an absolute path, so get the current directory path.
    theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "courses.xls";
    // Create a new Excel spreadsheet object and add the query data.
    theSheet = SpreadsheetNew("CourseData");
    SpreadsheetAddRows(theSheet, courses);
    // Define a format for the column.
    format1=StructNew();
    format1.font="Courier";
    format1.fontsize="10";
    format1.color="dark_blue;";
    format1.italic="true";
    format1.bold="true";
    format1.alignment="left";
    format1.textwrap="true";
    format1.fgcolor="pale_blue";
    format1.bottomborder="thick";
    format1.bottombordercolor="blue_grey";
    format1.topbordercolor="blue_grey";
    format1.topborder="thick";
    format1.leftborder="dotted";
    format1.leftbordercolor="blue_grey";
    format1.rightbordercolor="blue_grey";
    SpreadsheetFormatRow(theSheet, format1, "5");
</cfscript>

<!-- Write the spreadsheet to a file, replacing any existing file. -->
<cfspreadsheet action="write" filename="#theFile#" name="#theSheet#" sheet=1 sheetname="courses" overwrite=true>
SpreadsheetFormatRows

Description

Formats the contents of multiple rows of an Excel spreadsheet object.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

SpreadsheetFormatRows(spreadsheetObj, format, rows)

See also

SpreadsheetFormatCell, SpreadsheetFormatColumn, SpreadsheetFormatColumns, SpreadsheetFormatRow

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object in which to set the format.</td>
</tr>
<tr>
<td>format</td>
<td>A structure containing the format information. For details see SpreadsheetFormatColumn.</td>
</tr>
<tr>
<td>row</td>
<td>The rows to format, in one of the following formats:</td>
</tr>
<tr>
<td></td>
<td>• startRow-endRow - Insert rows in a single range.</td>
</tr>
<tr>
<td></td>
<td>• row, row, row... - Insert one or more individual rows.</td>
</tr>
<tr>
<td></td>
<td>You can also provide the formats together. For example, 1–5, 6, 7, 9–12.</td>
</tr>
</tbody>
</table>

Example

The following example creates a sheet, sets a format for rows 1, 3, and 5, and writes the result to a file:
<!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, CORLEVEL, COURSE_ID, CORNAME, CORDESC, LASTUPDATE
    FROM COURSELIST
</cfquery>

<cfscript>
///We need an absolute path, so get the current directory path.
theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "courses.xls";
//Create a new Excel spreadsheet object and add the query data.
theSheet = SpreadsheetNew("CourseData");
SpreadsheetAddRows(theSheet,courses);
/// Define a format for the column.
    format1=StructNew()
    format1.font="Courier";
    format1.fontsize="10";
    format1.color="dark_blue;";
    format1.italic="true";
    format1.bold="true";
    format1.alignment="left";
    format1.textwrap="true";
    format1.fgcolor="pale_blue";
    format1.bottomborder="thick";
    format1.bottombordercolor="blue_grey";
    format1.topbordercolor="blue_grey";
    format1.topborder="thick";
    format1.leftborder="dotted";
    format1.leftbordercolor="blue_grey";
    format1.rightbordercolor="blue_grey";
    SpreadsheetFormatRows(theSheet,format1,"1,3,5");
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheet=1 sheetname="courses" overwrite=true>
SpreadsheetGetCellComment

Description

Gets the comment for an Excel spreadsheet object cell as a structure with formatting information, or all comments for the object.

Returns

If the parameters include the row and column: a structure containing the comment information for the specified cell. If the function has only a spreadsheetObj parameter, an array containing a structure for each comment. Each structure has the following information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>A string containing the name of the comment author.</td>
</tr>
<tr>
<td>Column</td>
<td>The cell column number.</td>
</tr>
<tr>
<td>Comment</td>
<td>A string containing the comment text.</td>
</tr>
<tr>
<td>Row</td>
<td>The cell row number.</td>
</tr>
</tbody>
</table>

Category

Microsoft Office Integration

Function syntax

```plaintext
SpreadsheetGetCellComment(spreadsheetObj[, row, column])
```

See also

SpreadsheetFormatCell, SpreadsheetGetCellFormula, SpreadsheetGetCellValue, SpreadsheetMergeCells, SpreadsheetSetCellComment, SpreadsheetSetCellFormula, SpreadsheetSetCellValue

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object from which to get the comment.</td>
</tr>
<tr>
<td>row</td>
<td>The row number of the cell from which to get the comment.</td>
</tr>
</tbody>
</table>
The column number of the cell from which to get the comment.

Example

The following example sets and gets a comment for the cell at row 3 column 5.

```cfscript>
/// We need an absolute path, so get the current directory path.
theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "comment.xls";
// Create an Excel spreadsheet object.
theSheet = SpreadsheetNew();
// Define a cell comment.

comment1.anchor="1,1,15,20";
comment1.author="Adobe Systems";
comment1.bold="true";
comment1.color="lavender";
comment1.comment="This is the cell in row three, column 5 (E).";
comment1.fillcolor="yellow";
comment1.font="Courier";
comment1.horizontalignment="left";
comment1.linestyle="dashsys";
comment1.size="10";
comment1.verticalalignment="top";
// Set the comment.
SpreadsheetSetCellComment(theSheet,comment1,3,5);
// Get the comment from the Excel spreadsheet object.
theComment=SpreadsheetGetCellComment(theSheet,3,5);
</cfscript>
<cfoutput>
Row,Column: #theComment.row#, #theComment.column#<br />
Author: #theComment.author#<br />
Comment: #theComment.comment#
</cfoutput>
<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
sheet=1 sheetname="courses" overwrite=true>
SpreadsheetGetCellFormula

Description

Gets the formula for an Excel spreadsheet object cell, or all formulas for the object.

Returns

If the parameters include the row and column: a string containing the formula. If the function has the `spreadsheetObj` parameter, an array containing structures for each formula.

Category

Microsoft Office Integration

Function syntax

```plaintext
SpreadsheetGetCellFormula(spreadsheetObj[, row, column])
```

See also

`SpreadsheetGetCellComment`, `SpreadsheetFormatCell`, `SpreadsheetGetCellValue`, `SpreadsheetMergeCells`, `SpreadsheetSetCellComment`, `SpreadsheetSetCellFormula`, `SpreadsheetSetCellValue`

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>spreadsheetObj</code></td>
<td>The Excel spreadsheet object from which to get the formula.</td>
</tr>
<tr>
<td><code>row</code></td>
<td>The row number of the cell from which to get the formula.</td>
</tr>
<tr>
<td><code>column</code></td>
<td>The column number of the cell from which to get the formula.</td>
</tr>
</tbody>
</table>

Usage

If you specify only the `spreadsheetObj` parameter, the function returns an array of the structures with the following contents. The array has one entry for each cell that contains a formula.

<table>
<thead>
<tr>
<th>Field</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>formula</code></td>
<td>The formula for the cell.</td>
</tr>
<tr>
<td><code>row</code></td>
<td>The row number of the cell.</td>
</tr>
</tbody>
</table>
The column number of the cell.

Example

The following example sets a cell formula, and gets the cell formula and value.

```cfscript>
    //Create a new Excel spreadsheet object.
    theSheet=SpreadsheetNew();
    //Set the values of column 3 rows 1-10 to the row number.
    for (i=1; i<= 10; i=i+1)
        SpreadsheetSetCellValue(theSheet,i,i,3);
    //Set the formula for the cell in row 11 column 3 to be the sum of
    //Columns 1-10.
    SpreadsheetSetCellFormula(theSheet,"SUM(C1:C10)",11,3);
    //Get the formula from the Excel spreadsheet object.
    theValue=SpreadsheetGetCellFormula(theSheet,11,3);
    //Get the value of row 11 column 5 from the Excel spreadsheet object.
    theValue=SpreadsheetGetCellValue(theSheet,11,3);
</cfscript>

<cfoutput>
    Row 11, Column 3 value: #SpreadsheetGetCellValue(theSheet,11,3)#<br />
    Row 11, Column 3 formula: #SpreadsheetGetCellFormula(theSheet,11,3)#<br />
</cfoutput>
**SpreadsheetGetCellValue**

**Description**

Gets the Value for an Excel spreadsheet object cell.

**Returns**

A string containing the cell value.

**Category**

Microsoft Office Integration

**Function syntax**

```
SpreadsheetGetCellValue(spreadsheetObj, row, column)
```

**See also**

SpreadsheetGetCellComment, SpreadsheetFormatCell, SpreadsheetGetCellFormula, SpreadsheetMergeCells, SpreadsheetSetCellComment, SpreadsheetSetCellFormula, SpreadsheetSetCellValue

**History**

ColdFusion 9: Added the function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object from which to get the value.</td>
</tr>
<tr>
<td>row</td>
<td>The row number of the cell from which to get the formula.</td>
</tr>
<tr>
<td>column</td>
<td>The column number of the cell from which to get the formula.</td>
</tr>
</tbody>
</table>

**Example**

The following lines create an Excel spreadsheet object, set the value of the cell at row 3, column 5 - 365, gets the value and displays it:
<cfscript>
//Create a new Excel spreadsheet object.
theSheet=SpreadsheetNew();
//Set the value of the cell at row 3 column 5.
SpreadsheetSetCellValue(theSheet,365,3,5);
//Get the value from the Excel spreadsheet object.
theValue=SpreadsheetGetCellValue(theSheet,3,5);
WriteOutput("The value of column 5 row 3 is: " & theValue);
</cfscript>
SpreadsheetInfo

Description

Gets the property of an Excel spreadsheet object.

Returns

Returns a spreadsheet property that can be one of the following:

- AUTHOR
- CATEGORY
- COMMENTS
- CREATIONDATE
- LASTEDITED
- LASTAUTHOR
- LASTSAVED
- KEYWORDS
- MANAGER
- COMPANY
- SUBJECT
- TITLE
- SHEETS
- SHEETNAMES
- SPREADSHEETTYPES

Category

Microsoft Office Integration

Function syntax

SpreadsheetInfo(spreadsheetobj)

See also

SpreadsheetAddColumn, SpreadsheetAddImage, SpreadsheetAddRow, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheeobj</td>
<td>The Excel spreadsheet object from which to get the value.</td>
</tr>
</tbody>
</table>

Usage

This function is supported by Microsoft Office Excel 2007 and Microsoft Office 2003.
Example

```coldfusion
<cfspreadsheet action="read" src="#dirname#SingleSheet.xls" name="SpreadsheetObj"/>
<cfset info = SpreadSheetInfo(SpreadsheetObj)>
<cfoutput> AUTHOR : #info.author#<br> </cfoutput>
<cfoutput> Creation Date : #info.creationdate#<br> </cfoutput>
<cfoutput> LAST AUTHOR : #info.lastauthor#<br> </cfoutput>
<cfoutput> SHEETS : #info.sheets#<br> </cfoutput>
<cfoutput> SPREADSHEETTYPE : #info.SPREADSHEETTYPE#<br> </cfoutput>
<cfoutput> SUBJECT : #info.SUBJECT#<br> </cfoutput>
<cfoutput> TITLE : #info.TITLE#<br> </cfoutput>
```
SpreadsheetMergeCells

Description

Merges a rectangular block of two or more Excel spreadsheet object cells.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

SpreadsheetMergeCells(spreadsheetObj, startRow, endRow, startColumn, endColumn)

See also

SpreadsheetGetCellComment, SpreadsheetFormatCell, SpreadsheetGetCellFormula, SpreadsheetGetCellValue, SpreadsheetSetCellComment, SpreadsheetSetCellFormula, SpreadsheetSetCellValue

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object containing the cells to merge.</td>
</tr>
<tr>
<td>startRow</td>
<td>The number of the first row to merge.</td>
</tr>
<tr>
<td>endRow</td>
<td>The row number of the last row to merge.</td>
</tr>
<tr>
<td>startColumn</td>
<td>The column number of the first cell to merge.</td>
</tr>
<tr>
<td>endColumn</td>
<td>The column number of the last cell to merge.</td>
</tr>
</tbody>
</table>

Usage

If you merge two cells using this function, the merged cell by default displays the value in the cell that is on the left-hand side of the spreadsheet. For example, if you merge the cell (20,3) and cell (20,4), then the value in the cell (20, 3) is displayed. If the cell (20, 3) is blank, then after merging, the cell displays blank.

Example

The following example merges cells 4-6 in rows 1-3 of an Excel spreadsheet object. It puts text in the merged cells...
and saves the sheet to a file so you can see the result.

```cfscript
///We need an absolute path, so get the current directory path.
theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "mergecells.xls"

//Create a new Excel spreadsheet object and add the query data.
theSheet = SpreadsheetNew("CourseData");

//Merges cells 4-6 in the first three rows of the Excel spreadsheet object.
SpreadsheetMergeCells(theSheet,1,3,4,6);

//Set the value of the merged cell.
SpreadsheetSetCellValue(theSheet,"Columns 4-6 of rows 1-3 are merged",1,4);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet" overwrite=true>
SpreadsheetNew

Description

Creates a ColdFusionExcel spreadsheet object, which represents a single sheet of an Excel document.

Returns

ColdFusionExcel spreadsheet object.

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetNew([sheetName, xmlformat])
```

See also

All Other Spreadsheet functions: see Microsoft Office Integration list.

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sheetName</td>
<td>A string containing the sheet name to assign to the Excel spreadsheet object.</td>
</tr>
<tr>
<td>xmlformat</td>
<td>A Boolean value. True or Yes: Creates a .xlsx file that is supported by Microsoft Office Excel 2007. False or No: Creates a .xls file.</td>
</tr>
</tbody>
</table>

Usage

This function supports Microsoft Office Excel 2007. To create a simple .xls spreadsheet object with a default worksheet name, your code can be as follows:

```
<cfset SpreadsheetObj = spreadsheetNew()>
```

To create a simple .xls spreadsheet object by specifying the worksheet name as "mySheet", your code can be as follows:

```
<cfset SpreadsheetObj = spreadsheetNew("mySheet")>
```
To create spreadsheet objects that are supported by Microsoft Office Excel 2007 (.xlsx), your code can be as follows:

```coldfusion
<cfset SpreadsheetObj = spreadsheetNew("true")>

<cfset SpreadsheetObj = spreadsheetNew("mysheet","yes")>
```

**Note**

You can specify either "true" or "yes" to create a .xlsx file.

**Example**

The following example creates an Excel spreadsheet object with the sheet name Expenses, sets a cell value, and saves the result to a file.

```coldfusion
<cfscript>
///We need an absolute path, so get the current directory path.
    theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "newSpreadsheet.xls";
    //Create a new Excel spreadsheet object.
    theSheet = SpreadsheetNew("Expenses");
    //Set the value a cell.
    SpreadsheetSetCellValue(theSheet,"365",1,4);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet" overwrite=true>
SpreadsheetRead

Description

Reads a sheet from a spreadsheet file and stores it in a ColdFusion spreadsheet object.

Returns

Returns spreadsheet object.

Category

Microsoft Office Integration

Function syntax

```plaintext
SpreadSheetRead(fileName [, sheetName|sheet])
```

See also

SpreadsheetWrite, SpreadsheetAddColumn, SpreadsheetAddImage, SpreadsheetAddRow, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>A string specifying the path to the spreadsheet file.</td>
</tr>
<tr>
<td>sheetName</td>
<td>Optional parameter; name of the sheet. You can specify <code>sheet</code> or <code>sheetName</code>.</td>
</tr>
<tr>
<td>sheet</td>
<td>Optional parameter; number of the sheet. You can specify <code>sheet</code> or <code>sheetName</code>.</td>
</tr>
</tbody>
</table>

Usage

Use this function to read an Excel file with multiple sheets.

Example

```cfscript
{{<cfscript>
    a = SpreadSheetRead("C:\Files\Report.xls","Annual Report")
</cfscript>}}
```
SpreadsheetReadBinary

Description

Reads and stores content from a spreadsheet object and returns it as a byte array.

Returns

Returns a byte array of the stored spreadsheet information using the \texttt{cfcontent} tag.

Category

Microsoft Office Integration

Function syntax

\begin{verbatim}
SpreadsheetReadBinary(spreadsheetobj)
\end{verbatim}

See also

\texttt{SpreadsheetAddColumn}, \texttt{SpreadsheetAddImage}, \texttt{SpreadsheetAddRow}, \texttt{SpreadsheetDeleteRow}, \texttt{SpreadsheetDeleteRows}, \texttt{SpreadsheetFormatRow}, \texttt{SpreadsheetFormatRows}, \texttt{SpreadsheetShiftRows}

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{spreadSheetObject}</td>
<td>The Excel spreadsheet object to read.</td>
</tr>
</tbody>
</table>

Usage

Example

\begin{verbatim}
<cfheader name="Content-Disposition" value="inline; filename=test.xls">
<cfset a = spreadsheetnew()>
<cfset spreadsheetAddRow(a,"a,b,c")>
<!---You can do all the processing--->
<cfset bin = spreadsheetReadBinary(a)>
<cfcontent type="application/vnd-ms.excel" variable="#bin#" reset="true"/>
\end{verbatim}
SpreadsheetRemoveSheet

Description

Deletes a spreadsheet.

Returns

Nothing

Category

Microsoft Office Integration

Function syntax

SpreadsheetRemoveSheet (spreadsheetObj, sheetname)

See also

SpreadsheetSetActiveSheetNumber, SpreadsheetAddColumn, SpreadsheetAddImage, SpreadsheetAddRow, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9.0.1: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object from which you delete the sheet.</td>
</tr>
<tr>
<td>sheetname</td>
<td>Name of the sheet that must be removed.</td>
</tr>
</tbody>
</table>

Example

```cfc
<cfset spreadsheetVar= spreadsheetNew("New")>
<cfset spreadsheetCreateSheet(spreadsheetVar, "A")>
<cfset spreadsheetCreateSheet(spreadsheetVar, "B")>
<cfspreadsheet action="write" filename="#dirname#mySpreadSheet.xls" name="spreadsheetVar" overwrite="true" >
<cfspreadsheet action="read" src="#dirname#mySpreadSheet.xls" name="spreadsheetVar" >
<cfset spreadsheetRemoveSheet(spreadsheetVar, "B")>
<cfspreadsheet action="write" filename="#dirname#mySpreadSheet.xls" name="spreadsheetVar" overwrite="true" >
```
SpreadsheetSetActiveSheet

Description

Sets a specified sheet as active sheet.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```plaintext
SpreadsheetSetActiveSheet(spreadsheetobj, sheetname)
```

See also

- SpreadsheetSetActiveSheetNumber
- SpreadsheetAddColumn
- SpreadsheetAddImage
- SpreadsheetAddRow
- SpreadsheetDeleteRow
- SpreadsheetDeleteRows
- SpreadsheetFormatRow
- SpreadsheetFormatRows
- SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobj</td>
<td>The Excel spreadsheet object to which to add the split pane.</td>
</tr>
<tr>
<td>sheetname</td>
<td>The spreadsheet that must be set as active.</td>
</tr>
</tbody>
</table>

Usage

You can set only one sheet as active at a time. Therefore, any sheet operation is limited to a particular sheet. For operations in any other sheet, you must set that sheet as active.

Example

The following example shows how to switch from one sheet to another and perform operations.
<!--- Get the spreadsheet data as a query. --->
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, COURSE_ID, CORNAME
    FROM COURSELIST
</cfquery>

<cfscript>
/// We need an absolute path, so get the current directory path.
theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & courses.xls";
// Create a new Excel spreadsheet object and add the query data.
theSheet = SpreadsheetNew("CourseData");

SpreadsheetAddRows(theSheet, courses);
// Create a new sheet.
SpreadsheetCreateSheet (theSheet, "EvaluationSheet");
// Set the sheet as active.
SpreadsheetSetActiveSheet (theSheet, "EvaluationSheet");
// Add a new row to the sheet.
SpreadsheetAddRows(theSheet, courses);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="#theSheet#"
    sheet=1 sheetname="courses" overwrite=true>
SpreadsheetSetActiveSheetNumber

Description
Sets a specified sheetnumber as active sheet.

Returns
Does not return a value.

Category
Microsoft Office Integration

Function syntax

```
SpreadsheetSetActiveSheetNumber(spreadsheetobj, sheetnumber)
```

See also

SpreadsheetSetActiveSheet, SpreadsheetAddColumn, SpreadsheetAddImage, SpreadsheetAddRow, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftRows

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobj</td>
<td>The Excel spreadsheet object to which to add the split pane.</td>
</tr>
<tr>
<td>sheetnumber</td>
<td>The spreadsheet number that must be set as active.</td>
</tr>
</tbody>
</table>

Usage

You can set only one sheet as active at a time. Therefore, any sheet operation is limited to a particular sheet. For operations in any other sheet, you must set that sheet as active.

Example

The following example shows how to switch from one sheet to another and perform operations.
<cfquery
    name="courses" datasource="cfdocexamples"
    cachedwithin="#CreateTimeSpan(0, 6, 0, 0)#">
    SELECT CORNUMBER, DEPT_ID, COURSE_ID, CORNAME
    FROM COURSELIST
</cfquery>
<cfset theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & courses.xls">
<cfspreadsheet action="read" name=theSheet src="#theFile#" sheet="1">

<cfscript>
    SpreadsheetAddRows(theSheet, courses);
    // Set the sheet number 2 as active.
    SpreadsheetSetActiveSheetNumber (theSheet, 2);
    // Add a new row to the sheet 2.
    SpreadsheetAddRows(theSheet, courses);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet"
    sheet=1 sheetname="courses" overwrite=true>
SpreadsheetSetCellComment

Description

Specifies the comment for an Excel spreadsheet object cell.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

`SpreadsheetSetCellComment(spreadsheetObj, comment, row, column)`

See also

`SpreadsheetGetCellComment`, `SpreadsheetFormatCell`, `SpreadsheetGetCellFormula`, `SpreadsheetGetCellValue`, `SpreadsheetMergeCells`, `SpreadsheetSetCellFormula`, `SpreadsheetSetCellValue`

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object to which to add the comment.</td>
</tr>
<tr>
<td>comment</td>
<td>A structure containing the comment including text, formatting, and placement in the cell. See Usage for the structure contents.</td>
</tr>
<tr>
<td>row</td>
<td>The row number of the cell to which to add the comment.</td>
</tr>
<tr>
<td>column</td>
<td>The column number of the cell to which to add the comment.</td>
</tr>
</tbody>
</table>

Usage

The comment structure can have the following fields. Excel determines the default field values.

<table>
<thead>
<tr>
<th>Field</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Valid values</td>
</tr>
<tr>
<td>anchor</td>
<td>A comma separated list of integers specifying the position and size of the comment, in rows and columns, in the order top column, left row,bottom column, right row. For example: &quot;4,8,6,11&quot; specifies a comment with an upper left corner in row 4 column 8 and a lower right corner in row 6 column 11.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>author</td>
<td>The author's name.</td>
</tr>
<tr>
<td>bold</td>
<td>A Boolean value specifying whether the text is bold.</td>
</tr>
<tr>
<td>color</td>
<td>The text color, Any value in the Apache <a href="https://commons.apache.org/proper/commonspoi/apidocs/org/apache/poi/hssf/util/HSSFColor.html">org.apache.poi.hssf.util.HSSFColor class</a>: black, brown, olive_green, dark_green, dark_teal, dark_blue, indigo, grey_80_percent, orange, dark_yellow, green, teal, blue, blue_grey, grey_50_percent, red, light_orange, lime, sea_green, aqua, light_blue, violet, grey_40_percent, pink, gold, yellow, bright_green, turquoise, dark_red, sky_blue, plum, grey_25_percent, rose, light_yellow, light_green, light_turquoise, light_turquoise, pale_blue, lavender, white, cornflower_blue, lemon_chiffon, maroon, orchid, coral, royal_blue, light_cornflower_blue.</td>
</tr>
<tr>
<td>comment</td>
<td>A string containing the comment text.</td>
</tr>
<tr>
<td>fillcolor</td>
<td>A <a href="https://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Color.html">J2SE v1.4 java.awt.Color class</a> color value: white, lightGray, light_gray, gray, darkGray, dark_gray, black, red, pink, orange, yellow, green, magenta, cyan, blue. (Because ColdFusion is case independent, you do not need to specify the values if defined in the Java class.)</td>
</tr>
<tr>
<td>font</td>
<td>Any valid system font name.</td>
</tr>
<tr>
<td>horizontalalignment</td>
<td>The horizontal alignment of the text: left, center, right, justify, distributed.</td>
</tr>
<tr>
<td>italic</td>
<td>A Boolean value specifying whether the text is italic.</td>
</tr>
<tr>
<td>linestyle</td>
<td>The style of the top and right borders of the comment box: solid, dashsys, dotsys, dashdottsys, dashdotdottsys, dotgel, dashgel, longdashgel, dashdotgel, longdashdotgel, longdashdotdottgel.</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Linestylecolor</td>
<td>A Java color value (Does not work: BUG 72501).</td>
</tr>
<tr>
<td>Size</td>
<td>The size of the text in points.</td>
</tr>
<tr>
<td>Strikeout</td>
<td>A Boolean value specifying whether the text is struck out.</td>
</tr>
<tr>
<td>Underline</td>
<td>A Boolean value specifying whether the text is underlined.</td>
</tr>
<tr>
<td>Verticalalignment</td>
<td>The vertical alignment of the text: top, center, bottom, justify, distributed.</td>
</tr>
<tr>
<td>Visible</td>
<td>A Boolean value specifying whether the text is visible.</td>
</tr>
</tbody>
</table>

**Example**

The following example sets and gets a comment for the cell at row 3 column 5.
<cfscript>
///We need an absolute path, so get the current directory path.
   theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "comment.xls";
//Create an Excel spreadsheet object.
theSheet = SpreadsheetNew();
// Define a cell comment.
   comment1-structNew();
   comment1.anchor="0,0,5,8";
   comment1.author="Adobe Systems";
   comment1.bold="true";
   comment1.color="dark_green";
   comment1.comment="This is the cell in row three, column 5 (E).";
   comment1.fillcolor="light_gray";
   comment1.font="Courier";
   comment1.horizontallignment="left";
   comment1.linestyle="dashsys";
   comment1.size="10";
   comment1.verticalalignment="top";

   //Set the comment.
   SpreadsheetSetCellComment (theSheet,comment1,3,5);
   //Get the comment from the Excel spreadsheet object.
   theComment=SpreadsheetGetCellComment (theSheet,3,5);
</cfscript>

<coutput>
   Row,Column: #theComment.row#, #theComment.column#<br />
   Author: #theComment.author#<br />
   Comment: #theComment.comment#
</coutput>

<!---- Write the spreadsheet to a file, replacing any existing file. ---->
<cfspreadsheet action="write" filename="#theFile#" name="#theSheet" sheet=1 sheetname="courses" overwrite=true>
SpreadsheetSetCellFormula

Description

Specifies the formula for an Excel spreadsheet object cell.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetSetCellFormula(spreadsheetObj, formula, row, column)
```

See also

SpreadsheetGetCellComment, SpreadsheetFormatCell, SpreadsheetGetCellFormula, SpreadsheetGetCellValue, SpreadsheetMergeCells, SpreadsheetSetCellComment, SpreadsheetSetCellValue

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object to which to add the comment.</td>
</tr>
<tr>
<td>formula</td>
<td>A string containing the formula.</td>
</tr>
<tr>
<td>row</td>
<td>The row number of the cell to which to add the formula.</td>
</tr>
<tr>
<td>column</td>
<td>The column number of the cell to which to add the formula.</td>
</tr>
</tbody>
</table>

Usage

This function replaces any existing value, including specific entered values.

Example

The following line sets the formula for the cell at row 2 column 11 to be the sum of the cells in the column's rows 1 through 12. The following example sets a cell formula, and gets the cell formula and value.
<cfscript>
    //Create a new Excel spreadsheet object.
    theSheet=SpreadsheetNew();
    //Set the values of column 3 rows 1-10 to the row number.
    for (i=1; i<=10; i=i+1)
        SpreadsheetSetCellValue(theSheet,i,i,3);
    //Set the formula for the cell in row 11 column 3 to be the sum of
    //Columns 1-10.
    SpreadsheetSetCellFormula(theSheet,"SUM(C1:C10)",11,3);
    //Get the formula from the Excel spreadsheet object.
    theValue=SpreadsheetGetCellFormula(theSheet,11,3);
    //Get the value of row 11 column 5 from the Excel spreadsheet object.
    theValue=SpreadsheetGetCellValue(theSheet,11,3);
</cfscript>

<cfoutput>
    Row 11, Column 3 value: #SpreadsheetGetCellValue(theSheet,11,3)#<br />
    Row 11, Column 3 formula: #SpreadsheetGetCellFormula(theSheet,11,3)#<br />
</cfoutput>
SpreadsheetSetCellValue

Description

Specifies the value of an Excel spreadsheet object cell.

Category

Microsoft Office Integration

Function syntax

SpreadsheetSetCellValue(spreadsheetObj, value, row, column, datatype)

See also

SpreadsheetGetCellComment, SpreadsheetFormatCell, SpreadsheetGetCellFormula, SpreadsheetGetCellValue, SpreadsheetMergeCells, SpreadsheetSetCellComment, SpreadsheetSetCellFormula

History

ColdFusion 11: Added the datatype attribute.

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object to which to add the comment.</td>
</tr>
<tr>
<td>value</td>
<td>A string containing the cell value.</td>
</tr>
<tr>
<td>row</td>
<td>The row number of the cell to which to set the value.</td>
</tr>
<tr>
<td>column</td>
<td>The column number of the cell to which to set the value.</td>
</tr>
<tr>
<td>datatype</td>
<td>The data type denote by the cell. The allowed values are NUMERIC, DATE, and STRING.</td>
</tr>
</tbody>
</table>

Example

The following lines create an Excel spreadsheet object, set the value of the cell at row 3, column 5 to 365, and get the value:
<cfscript>
  //Create a new Excel spreadsheet object.
  theSheet=SpreadsheetNew();
  //Set the value of the cell at row 3 column 5.
  SpreadsheetSetCellValue(theSheet,365,3,5);
  //Get the value from the Excel spreadsheet object.
  theValue=SpreadsheetGetCellValue(theSheet,3,5);
  WriteOutput("The value of column 5 row 3 is: " & theValue);
</cfscript>
SpreadsheetSetColumnWidth

Description
Sets the width of a column in a worksheet.

Returns
Nothing

Category
Microsoft Office Integration

Function syntax

SpreadSheetSetColumnWidth(spreadsheetobj, column number, width)

See also
SpreadsheetGetCellComment, SpreadsheetFormatCell, SpreadsheetGetCellFormula, SpreadsheetGetCellValue, SpreadsheetMergeCells, SpreadsheetSetCellFormula, SpreadsheetSetCellValue

History
ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobj</td>
<td>The Excel spreadsheet object to which to set the column width.</td>
</tr>
<tr>
<td>column number</td>
<td>Specifies the column to set the width. The</td>
</tr>
<tr>
<td>width</td>
<td>Specifies the width in points.</td>
</tr>
</tbody>
</table>

Example
The following example creates a spreadsheet, adds columns to the spreadsheet, and sets the column width for the newly added columns.
<cfscript>
a=SpreadSheetNew();
SpreadSheetAddRow(a, "1,2,3,4,5,6,7,8");
SpreadSheetAddRow(a, "1,2,3,4,5,6,7,8", 2, 1);
</cfscript>
<cfset SpreadSheetSetColumnWidth(a, 2, 10)>
<cfset SpreadSheetSetColumnWidth(a, 3, 25)>
<cfspreadsheet action="write" filename="#expandpath('.')#/b.xls" name="a"
overwrite="true">
SpreadsheetSetFooter

Description

Adds a footer to the specified worksheet.

Category

Microsoft Office Integration

Function syntax

SpreadsheetSetFooter(spreadsheetobj, left footer, center footer, right footer)

See also

SpreadsheetGetCellComment, SpreadsheetFormatCell, SpreadsheetGetCellFormula, SpreadsheetGetCellValue, SpreadsheetMergeCells, SpreadsheetSetCellFormula, SpreadsheetSetCellValue

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobj</td>
<td>The Excel spreadsheet object to which to add the footer.</td>
</tr>
<tr>
<td>left footer</td>
<td>Adds the footer in the left side of the worksheet.</td>
</tr>
<tr>
<td>center footer</td>
<td>Adds the footer in the center of the worksheet.</td>
</tr>
<tr>
<td>right footer</td>
<td>Adds the footer in the right side of the worksheet.</td>
</tr>
</tbody>
</table>

Usage

The footer that you add appears on the printed worksheet document.

Example

The following example adds a footer in the left side of the worksheet.
{{<cfspreadsheet action="read" src="#dirname#SingleSheet.xls" sheet="2"
name="SpreadsheetObj" >
<cfset spreadsheetSetHeader(SpreadsheetObj,"left header","center header","right
header")>
<cfset spreadsheetSetFooter(SpreadsheetObj,"left footer","center footer","right
footer")>
<cfspreadsheet action="write" filename="#dirname#MySheet.xls" name="SpreadsheetObj"
overwrite="true" >}}
SpreadsheetSetHeader

Description

Adds a header to the specified worksheet.

Returns

Nothing

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetSetHeader(spreadsheetobj, left header, center header, right header)
```

See also

SpreadsheetGetCellComment, SpreadsheetFormatCell, SpreadsheetGetCellFormula, SpreadsheetGetCellValue, SpreadsheetMergeCells, SpreadsheetSetCellFormula, SpreadsheetSetCellValue

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobj</td>
<td>The Excel spreadsheet object to which to add the header.</td>
</tr>
<tr>
<td>left header</td>
<td>Adds the header in the left side of the worksheet.</td>
</tr>
<tr>
<td>center header</td>
<td>Adds the header in the center of the worksheet.</td>
</tr>
<tr>
<td>right header</td>
<td>Adds the header in the right side of the worksheet.</td>
</tr>
</tbody>
</table>

Usage

The header that you add appears on the printed worksheet document.

Example

The following example adds a header in the center of the worksheet.
{{<cfspreadsheet action="read" src="#dirname#Singleton.xls" sheet="2" name="SpreadsheetObj" ><cfset spreadsheetSetHeader(SpreadsheetObj,"left header","center header","right header")><cfset spreadsheetSetFooter(SpreadsheetObj,"left footer","center footer","right footer")><cfspreadsheet action="write" filename="#dirname#MySheet.xls" name="SpreadsheetObj" overwrite="true" >}}
SpreadsheetSetRowHeight

Description

Sets the height of a row in a worksheet.

Returns

Nothing

Category

Microsoft Office Integration

Function syntax

SpreadSheetSetRowHeight(spreadsheetobj, row number, height)

See also

SpreadsheetGetCellComment, SpreadsheetFormatCell, SpreadsheetGetCellFormula, SpreadsheetGetCellValue, SpreadsheetMergeCells, SpreadsheetSetCellFormula, SpreadsheetSetCellValue

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetobj</td>
<td>The Excel spreadsheet object to which to set the column width.</td>
</tr>
<tr>
<td>row number</td>
<td>Specifies the row to set the height.</td>
</tr>
<tr>
<td>height</td>
<td>Specifies the height in points.</td>
</tr>
</tbody>
</table>

Usage

Example

The following example creates a spreadsheet, adds rows to the spreadsheet, and sets the row height for the newly added rows.
<cfscript>
a=SpreadSheetNew();
SpreadSheetAddRow(a,"1,2,3,4,5,6,7,8");
SpreadSheetAddRow(a,"1,2,3,4,5,6,7,8",2,1);
</cfscript>
<cfset SpreadSheetSetRowHeight(a,2,10)>
<cfset SpreadSheetSetRowHeight(a,3,25)>
<cfspreadsheet action="write" filename="#expandpath('.')#/b.xls" name="a" overwrite="true">
SpreadsheetShiftColumns

Description
Shifts one or more columns in Excel spreadsheet object left or right.

Returns
Does not return a value.

Category
Microsoft Office Integration

Function syntax

<table>
<thead>
<tr>
<th>Function syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpreadsheetShiftColumns(spreadsheetObj, start[, cols])</td>
</tr>
<tr>
<td>SpreadsheetShiftColumns(spreadsheetObj, start, end [, cols])</td>
</tr>
</tbody>
</table>

See also
SpreadsheetAddColumn, SpreadsheetDeleteColumn, SpreadsheetDeleteColumns, SpreadsheetFormatColumn, SpreadsheetFormatColumns, SpreadsheetShiftRows

History
ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object in which to make the shift.</td>
</tr>
<tr>
<td>start</td>
<td>The number of the first, or only, column to shift</td>
</tr>
<tr>
<td>end</td>
<td>The number of the last column to shift. If you omit this parameter, the function shifts a single column.</td>
</tr>
<tr>
<td>columns</td>
<td>The positive (right) or negative (left) number of columns by which to shift the columns. If you omit this parameter, the function shifts the column right by one unit.</td>
</tr>
</tbody>
</table>

Usage

Example
The following line shifts columns 6 and 7 two columns to the left.
<cfset SpreadsheetShiftColumns(SpreadsheetObj,10,11,2)>  
// We need an absolute path, so get the current directory path.
   theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "shiftcolumns.xls";
// Create a new Excel spreadsheet object.
   theSheet = SpreadsheetNew("Expenses");
// Set some cell values, indicating their initial location.
   SpreadsheetSetCellValue(theSheet,"Cell D10",10,4);
   SpreadsheetSetCellValue(theSheet,"Cell E12",12,5);
   SpreadsheetSetCellValue(theSheet,"Cell F12",12,6);
   SpreadsheetSetCellValue(theSheet,"Cell G13",13,7);
// Shift columns 6 and 7 left 2 columns.
   SpreadsheetShiftColumns(theSheet,6,7,-2);
</cfscript>

<!--- Write the spreadsheet to a file, replacing any existing file. --->
<cfspreadsheet action="write" filename="#theFile#" name="theSheet" overwrite=true>
SpreadsheetShiftRows

Description

Shifts one or more rows in Excel spreadsheet object up or down. The contents of the shifted row, including empty cells, overwrites data in the column to which it is shifted.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```
SpreadsheetShiftRows(spreadsheetObj, start[, rows])
SpreadsheetShiftRows(spreadsheetObj, start, end, rows)
```

See also

SpreadsheetAddRow, SpreadsheetAddRows, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftColumns

History

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object in which to make the shift.</td>
</tr>
<tr>
<td>start</td>
<td>The number of the first row to shift</td>
</tr>
<tr>
<td>end</td>
<td>The number of the last row to shift. If you omit this parameter, the function shifts a single row.</td>
</tr>
<tr>
<td>rows</td>
<td>The positive (down) or negative (up) number of rows by which to shift the rows. If you omit this parameter, the function shifts the row down by one unit.</td>
</tr>
</tbody>
</table>

Usage

Example

The following line shifts 10 and 11 down two rows. Notice that the shifted rows completely overwrite the previous rows 12 and 13.
<cfscript>
  /// We need an absolute path, so get the current directory path.
  theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "shiftrows.xls";
  // Create a new Excel spreadsheet object.
  theSheet = SpreadsheetNew("Expenses");
  // Set some cell values, indicating their initial location.
  SpreadsheetSetCellValue(theSheet,"Cell D10",10,4);
  SpreadsheetSetCellValue(theSheet,"Cell E11",11,5);
  SpreadsheetSetCellValue(theSheet,"Cell A12",12,1);
  SpreadsheetSetCellValue(theSheet,"Cell B13",13,2);
  // Shift rows 10 and 11 down 2 rows.
  SpreadsheetShiftRows(theSheet,10,11,2);
</cfscript>

<cfspreadsheet action="write" filename="#theFile#" name="theSheet" overwrite=true>
SpreadsheetWrite

Description

Writes single sheet to a new XLS file from a ColdFusion spreadsheet object.

Category

Microsoft Office Integration

Function syntax

```plaintext
SpreadSheetWrite(SpreadsheetObj, fileName)
SpreadSheetWrite(SpreadsheetObj, fileName [, overwrite])
SpreadSheetWrite(SpreadsheetObj, fileName [, password])
SpreadSheetWrite(SpreadsheetObj, fileName [, password, overwrite])
SpreadSheetWrite(SpreadsheetObj, fileName [, password, overwrite, autosize])
```

See also

SpreadsheetRead, SpreadsheetAddRow, SpreadsheetAddRows, SpreadsheetDeleteRow, SpreadsheetDeleteRows, SpreadsheetFormatRow, SpreadsheetFormatRows, SpreadsheetShiftColumns

History

ColdFusion 11: Added the autosize attribute.

ColdFusion 9: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object to which to write.</td>
</tr>
<tr>
<td>fileName</td>
<td>The pathname of the file that is written.</td>
</tr>
<tr>
<td>overwrite</td>
<td>A Boolean value specifying whether to overwrite an existing file. Specify yes to overwrite.</td>
</tr>
<tr>
<td>password</td>
<td>Password to protect the active sheet. Password is applicable only for Excel 97 - 2003 file formats. It will be ignored for XML file format (Excel 2007).</td>
</tr>
<tr>
<td>autosize</td>
<td>Auto re-size the column, if required. You can specify either a Boolean value or an array of integers denoting the columns to expand. If you specify an array, it must contain the column numbers that needs to be re-sized.</td>
</tr>
</tbody>
</table>

Usage
Use this function to:

- Write multiple sheets to a single file.
- Update an existing file, read all sheets in the file, modify one or more sheets, and to rewrite the entire file.

Example

```cfs
<cfscript>
spreadsheet = SpreadsheetRead("C:\Files\Report.xls","Annual Report");
SpreadSheetWrite(spreadsheet,"C:\Files\Report.xls","P@ssword","yes");
</cfscript>
```

Example 2

```cfs
<cfscript>
spObj = spreadsheetread("#dirname#SingleSheet.xls","Sheet2");
spreadsheetCreateSheet(spObj,"A");
spreadsheetaddrow(spObj,"x,x,x,x,x",3,1);
spreadsheetsetActiveSheet(spObj,"A");
spreadsheetaddrow(spObj,"z,z,z,z",3,1);
spreadsheetsetActiveSheetNumber(spObj,1);
spreadsheetaddrow(spObj,"a,b,c,d,e",3,1);
SpreadsheetWrite(spObj,"#dirname#SingleSheet1.xls","yes");
</cfscript>
```

Example 3
<!--- Using the autosize param.  
SpreadSheetWrite method --->

```
<cfset colList ="col1,col2,col3,col4,col5,col6,col7,col8,col9,col10">
<cfset rowCount = 100>
<cfset qryObj = QueryNew("#colList#")>
<cfset QueryAddRow(qryObj, #rowCount#)>
<cfloop from="1" to="#rowCount#" index="r">
    <cfloop from="1" to="#ListLen(colList)#" index="c">
        <cfset QuerySetCell(qryObj, #ListGetAt(colList,c)#, "some random text r#r# c#c#", r)>
    </cfloop>
</cfloop>

<cfset xlObj = spreadsheetNew("testsheet", true)>
<cfset spreadsheetAddRows(xlObj, "#qryObj#")>
<cfset spreadsheetwrite(xlObj, "Expandpath(./)\#test_autosize_ss_method.xlsx", "", true, false)>
```
### Sqr

**Description**

Calculates the square root of a number.

**Returns**

Number; square root of `number`.

**Category**

*Mathematical functions*

**Function syntax**

```
Sqr(number)
```

**See also**

*Abs*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>number</code></td>
<td>A positive integer or a variable that contains one. Number whose square root to get.</td>
</tr>
</tbody>
</table>

**Usage**

The value in `number` must be greater than or equal to 0.

**Example**

```html
<h3>Sqr Example</h3>

<p>Returns the square root of a positive number.</p>

<p>Sqr(2): <cfoutput>#Sqr(2)#</cfoutput></p>
<p>Sqr(Abs(-144)): <cfoutput>#Sqr(Abs(-144))#</cfoutput></p>
<p>Sqr(25^2): <cfoutput>#Sqr(25^2)#</cfoutput></p>
```
StripCR

Description

Deletes return characters from a string.

Returns

A copy of string, after removing return characters.

Category

Display and formatting functions, String functions

Function syntax

StripCR(string)

See also

ParagraphFormat

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

Usage

Useful for preformatted (between <pre> and </pre> tags) HTML display of data entered in textarea fields.

Example
<h3>StripCR Example</h3>

<p>Function StripCR is useful for preformatted HTML display of data (PRE) entered in textarea fields. <cfif isdefined("Form.myTextArea")>
</cfif>
</pre>
<cfoutput> StripCR(Form.myTextArea) </cfoutput>
</pre>
</cfif>

<!--- use #Chr(10)#Chr(13)# to simulate line feed/carriage return combination --->
<form action = "stripcr.cfm">
<textarea name = "MyTextArea" cols = "35" rows = 8>
This is sample text and you see how it scrolls
 From one line
 to the next
</textarea>
<input type = "Submit" name = "Show me the HTML version">
</form>
**StructAppend**

**Description**

Appends one structure to another.

**Returns**

True, upon successful completion; False, otherwise.

**Category**

Structure functions

**Function syntax**

```
StructAppend(struct1, struct2, overwriteFlag)
```

**See also**

Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

**History**

ColdFusion MX: Changed behavior: this function can be used on XML objects.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct1</td>
<td>Structure to which struct2 is appended.</td>
</tr>
<tr>
<td>struct2</td>
<td>Structure that contains the data to append to struct1</td>
</tr>
</tbody>
</table>
| overwriteFlag| • True or Yes: values in struct2 overwrite corresponding values in struct1. Default.  
• False or No: values in struct2 do not overwrite corresponding values in struct1. |

**Usage**

This function appends the fields and values of struct2 to struct1; struct2 is not modified. If struct1 already contains a field of struct2, overwriteFlag determines whether the value in struct2 overwrites it. A structure's keys are unordered.

**Example**
<html>
<body>
<!---- Create a Name structure --->
<cfset nameCLK=StructNew()>
<cfset nameCLK.first="Chris">
<cfset nameCLK.middle="Lloyd">
<cfset nameCLK.last="Gilson">
<!---- Create an address struct --->
<cfset addrCLK=StructNew()>
<cfset addrCLK.street="17 Gigantic Rd">
<cfset addrCLK.city="Watertown">
<cfset addrCLK.state="MA">
<cfset addrCLK.zip="02472">
<!---- Create a Person structure --->
<cfset personCLK=StructNew()>
<cfset personCLK.name=#nameCLK#>
<cfset personCLK.addr=#addrCLK#>
<!---- Display the contents of the person struct before the Append --->
<p>
The person struct <b>before</b> the Append call:<br>
<cfloop collection=#personCLK# item="myItem">
<cfoutput>
#myItem#<br>
</cfoutput>
</cfloop>
<!---- Merge the address struct into the top-level person struct --->
<cfset bSuccess = StructAppend( personCLK, addrCLK )>
<!---- Display the contents of the person struct, after the Append --->
<p>
The person struct <b>after</b> the Append call:<br>
<cfloop collection=#personCLK# item="myItem">
<cfoutput>
#myItem#<br>
</cfoutput>
</cfloop>
StructClear

Description

Removes all data from a structure.

Returns

True, on successful execution; False, otherwise.

Category

Structure functions

Function syntax

```
StructClear(structure)
```

See also

Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

History

ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure</td>
<td>Structure to clear</td>
</tr>
</tbody>
</table>

Usage

Do not call this function on a session variable. For more information, see TechNote, "ColdFusion 4.5 and the StructClear(Session) function," at go.adobe.com/kb/ts_tn_17479_en-us. (The article applies to ColdFusion 4.5, 5.x, and ColdFusion MX.)

Example
<body>
<h1>Add New Employees</h1>
<cfparam name = "Form.firstname" default = "">
<cfparam name = "Form.lastname" default = "">
<cfparam name = "Form.email" default = "">
<cfparam name = "Form.phone" default = "">
<cfparam name = "Form.department" default = "">
<cfif form.firstname eq "">
<p>Please fill out the form.</p>
<cfelse>
<cfoutput>
<cfscript>
employee = StructNew();
StructInsert(employee, "firstname", Form.firstname);
StructInsert(employee, "lastname", Form.lastname);
StructInsert(employee, "email", Form.email);
StructInsert(employee, "phone", Form.phone);
StructInsert(employee, "department", Form.department);
</cfscript>
</cfoutput>
<cfif form.firstname eq "">
<cf_addemployee empinfo = "#employee#"> StructClear(employee);</cfscript>
</cfif>
</cfoutput>
</body>
StructCopy

Description

Copies a structure. Copies top-level keys, values, and arrays in the structure by value; copies nested structures by reference.

Returns

A copy of a structure, with the same keys and values; if structure does not exist, throws an exception.

Category

Structure functions

Function syntax

```
StructCopy(structure)
```

See also

Structure functions; Structure functions in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure</td>
<td>Structure to copy</td>
</tr>
</tbody>
</table>

Usage

The following code shows how this function copies a structure that contains a string field, a number field, and a two-dimensional array at the top-level:

```
<cfoutput>
   <cfset assignedCopy = StructNew()>
   <cfset assignedCopy.string = #struct.string#>
   <cfset assignedCopy.number = #struct.number#>
   <cfset assignedCopy.array = ArrayNew(2)>
   <cfset assignedCopy.array[1][1] = #struct.array[1][1]#>
   <cfset assignedCopy.array[1][2] = #struct.array[1][2]#>
</cfoutput>
```

The following code shows how StructCopy copies a nested structure:

```
<cfoutput>
   <cfset assignedCopy.nestedStruct = struct.nestedStruct>
</cfoutput>
```
To copy a structure entirely by value, use **Duplicate**. The following table shows how variables are assigned:

<table>
<thead>
<tr>
<th>Variable type</th>
<th>Assigned by</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure.any_simple_value</td>
<td>Binary</td>
</tr>
<tr>
<td>structure.array</td>
<td>Value</td>
</tr>
<tr>
<td>structure.nested_structure</td>
<td>Reference</td>
</tr>
<tr>
<td>structure.object</td>
<td>Reference</td>
</tr>
<tr>
<td>structure.query</td>
<td>Reference</td>
</tr>
</tbody>
</table>

**Example**

```cfc
<!--- This code shows assignment by-value and by-reference. --->
// This script creates a structure that StructCopy copies by value. <br>
<cfscript>
// Create elements.
  s = StructNew();
  s.array = ArrayNew(2);

  // Assign simple values to original top-level structure fields.
  s.number = 99;
  s.string = "hello tommy";

  // Assign values to original top-level array.
  s.array[1][1] = "one one";
  s.array[1][2] = "one two";
</cfscript>

<!--- Output original structure --->
<hr>
<b>Original Values</b><br>
<cfoutput>
  // Simple values <br>
  s.number = #s.number#<br>
  s.string = #s.string#<br>

  // Array value <br>
  s.array[1][1] = #s.array[1][1]#<br>
  s.array[1][2] = #s.array[1][2]#<br>
</cfoutput>

// Copy this structure to a new structure. <br>
<cfset copied = StructCopy(s)>

<cfscript>
// Change the values of the original structure. <br>
  s.number = 100;
  s.string = "hello tommy (modified)";
  s.array[1][1] = "one one (modified)";
  s.array[1][2] = "one two (modified)";
</cfscript>
```
<hr>
<b>Modified Original Values</b><br>
<cfoutput>
// Simple values <br>
s.number = #s.number#<br>
s.string = #s.string#<br>
// Array value <br>
s.array[1][1] = #s.array[1][1]#<br>
s.array[1][2] = #s.array[1][2]#<br>
</cfoutput>
<hr>
<b>Copied structure values should be the same as the original.</b><br>
<cfoutput>
// Simple values <br>
copied.number = #copied.number#<br>
copied.string = #copied.string#<br>
// Array value <br>
copied.array[1][1] = #copied.array[1][1]#<br>
copied.array[1][2] = #copied.array[1][2]#<br>
</cfoutput>

// This script creates a structure that StructCopy copies by reference.
<cfscript>
// Create elements.
s = StructNew();
s.nested = StructNew();
s.nested.array = ArrayNew(2);
// Assign simple values to nested structure fields.
s.nested.number = 99;
s.nested.string = "hello tommy";
// Assign values to nested array.
s.nested.array[1][1] = "one one";
s.nested.array[1][2] = "one two";
</cfscript>

<!--- Output original structure --->
<hr>
<b>Original Values</b><br>
<cfoutput>
// Simple values <br>
s.nested.number = #s.nested.number#<br>
s.nested.string = #s.nested.string#<br>
// Array values <br>
s.nested.array[1][1] = #s.nested.array[1][1]#<br>
s.nested.array[1][2] = #s.nested.array[1][2]#<br>
</cfoutput>

// Use StructCopy to copy this structure to a new structure. <br>
<cfset copied = StructCopy(s)>
// Use Duplicate to clone this structure to a new structure. <br>
<cfset duplicated = Duplicate(s)>

<cfscript>
// Change the values of the original structure.
s.nested.number = 100;
s.nested.string = "hello tommy (modified)";
s.nested.array[1][1] = "one one (modified)";
s.nested.array[1][2] = "one two (modified)";
</cfscript>
</cfscript>
<br>
<b>Modified Original Values</b><br>
<cfoutput>
// Simple values <br>
s.nested.number = #s.nested.number#<br>
s.nested.string = #s.nested.string#<br>

// Array value <br>
s.nested.array[1][1] = #s.nested.array[1][1]#<br>
s.nested.array[1][2] = #s.nested.array[1][2]#<br>
</cfoutput>
<br>
<b>Copied structure values should reflect changes to original.</b><br>
<cfoutput>
// Simple values <br>
copied.nested.number = #copied.nested.number#<br>
copied.nested.string = #copied.nested.string#<br>
// Array values <br>
copied.nested.array[1][1] = #copied.nested.array[1][1]#<br>
copied.nested.array[1][2] = #copied.nested.array[1][2]#<br>
</cfoutput>
<br>
<b>Duplicated structure values should remain unchanged.</b><br>
<cfoutput>
// Simple values <br>
duplicated.nested.number = #duplicated.nested.number#<br>
duplicated.nested.string = #duplicated.nested.string#<br>
// Array value <br>
duplicated.nested.array[1][1] = #duplicated.nested.array[1][1]#<br>
</cfoutput>
duplicated.nested.array[1][2] = #duplicated.nested.array[1][2]#<br>
</cfoutput>
**StructCount**

**Description**

Counts the keys in a structure.

**Returns**

A number; if `structure` does not exist, throws an exception.

**Category**

*Structure functions*

**Function syntax**

```plaintext
StructCount(structure)
```

**See also**

*Structure functions; Modifying a ColdFusion XML object* in the *Developing ColdFusion Applications*

**History**

ColdFusion MX: Changed behavior: this function can be used on XML objects.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>structure</code></td>
<td>Structure to access</td>
</tr>
</tbody>
</table>

**Example**
This view-only example shows use of StructCount. This file is similar to addemployee.cfm, which is called by StructNew, StructClear, and StructDelete. To test, copy StructCount function to appropriate place in addemployee.cfm.

```cfc
cfswitch expression = "#ThisTag.ExecutionMode#"
cfcase value = "start"
  cftime StructIsEmpty(attributes.EMPINFO)
  cfooutput>Error. No employee data was passed.</cfooutput>
  cfexit method = "ExitTag"
  cfelse
  cquery name = "AddEmployee" datasource = "cfdocexamples"
  INSERT INTO Employees
  (FirstName, LastName, Email, Phone, Department)
  VALUES
  (  
    ' #StructFind(attributes.EMPINFO, "firstname")#' ,
    ' #StructFind(attributes.EMPINFO, "lastname")#',
    ' #StructFind(attributes.EMPINFO, "email")#',
    ' #StructFind(attributes.EMPINFO, "phone")#',
    ' #StructFind(attributes.EMPINFO, "department")#'
  )
  cfooutput><hr>Employee Add Complete  
  cfooutput><p>#StructCount(attributes.EMPINFO)# columns added.</cfooutput>
  cfcase
  cfend
  cfend
```
StructDelete

Description

Removes an element from a structure.

Returns

Boolean value. The value depends on the indicatenotexisting parameter value.

Category

Structure functions

Function syntax

```
StructDelete(structure, key [, indicatenotexisting ])
```

See also

Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

History

ColdFusion MX: Changed behavior: this function can be used on XML objects.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure</td>
<td>Structure or a variable that contains one. Contains element to remove.</td>
</tr>
<tr>
<td>key</td>
<td>Element to remove.</td>
</tr>
<tr>
<td>indicatenotexisting</td>
<td>• True: returns Yes if key exists; No if it does not.</td>
</tr>
<tr>
<td></td>
<td>• False: returns Yes regardless of whether key exists. Default.</td>
</tr>
</tbody>
</table>

Example
<h3>StructDelete Function</h3>

<!-- Delete the surrounding comments to make this page work
<p>This example uses the StructInsert and StructDelete functions.
<!-- Establish params for first time through --->
<cfparam name = "firstname" default = "Mary">
<cfparam name = "lastname" default = "Sante">
<cfparam name = "email" default = "msante@allaire.com">
<cfparam name = "phone" default = "777-777-7777">
<cfparam name = "department" default = "Documentation">

<cfif IsDefined("FORM.Delete")>
  <cfoutput>
    Field to be deleted: #form.field#
  </cfoutput>
</cfif>

<CFScript>
  employee = StructNew();
  StructInsert(employee, "firstname", firstname);
  StructInsert(employee, "lastname", lastname);
  StructInsert(employee, "email", email);
  StructInsert(employee, "phone", phone);
  StructInsert(employee, "department", department);
</CFScript>

Before deletion, employee structure looks like this:
<cfdump var="#employee#">
<br>
<cfset rc = StructDelete(employee, ":form.field:", "True")>
<cfoutput>
  Did I delete the field ":form.field:"? The code indicates: #rc#<br>
  The structure now looks like this:<br>
</cfoutput>
<cfdump var="#employee#">
<br>
</cfoutput>
</cfif>
<br>
<form method="post" action="#CGI.Script_Name#">
  Select the field to be deleted:<br>
  <select name = "field">
    <option value = "firstname">first name
    <option value = "lastname">last name
    <option value = "email">email
    <option value = "phone">phone
    <option value = "department">department
  </select>
  <input type = "submit" name = "Delete" value = "Delete">
</form>
Delete this comment to make this page work --->
StructEach

Description
Used to loop over elements in a structure by accessing key-value pairs.

Returns
Nothing

Category
Closure functions

Syntax
structEach(struct, function(key, value) {});

See also
Other closure functions.

History
ColdFusion 10: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct</td>
<td>Name of the structure object.</td>
</tr>
<tr>
<td>function</td>
<td>Inline function executed for each key - value pair in the struct.</td>
</tr>
<tr>
<td>key</td>
<td>Key in a struct.</td>
</tr>
<tr>
<td>value</td>
<td>Value in a struct.</td>
</tr>
</tbody>
</table>
**StructFilter**

**Description**

Used to filter the key value pairs in a struct.

**Returns**

A new struct

**Category**

Closure functions

**Syntax**

```coldfusion
structFilter(struct, function(key, value){return true|false;});
```

**See also**

Other closure functions.

**History**

ColdFusion 10: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct</td>
<td>Name of the struct object.</td>
</tr>
<tr>
<td>function</td>
<td>Inline function executed for each element in the array. Returns true if the key value pair in the struct has to be included in the resultant struct.</td>
</tr>
<tr>
<td>key</td>
<td>Key in a struct.</td>
</tr>
<tr>
<td>value</td>
<td>Value in a struct.</td>
</tr>
</tbody>
</table>
**StructFind**

**Description**

Determines the value associated with a key in a structure.

**Returns**

The value associated with a key in a structure; if `structure` does not exist, throws an exception.

**Category**

*Structure functions*

**Function syntax**

```
StructFind(structure, key)
```

**See also**

*Structure functions: Structure functions* in the *Developing ColdFusion Applications*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>structure</code></td>
<td>Structure that contains the value to return</td>
</tr>
<tr>
<td><code>key</code></td>
<td>Key whose value to return</td>
</tr>
</tbody>
</table>

**Usage**

A structure’s keys are unordered.

**Example**

```
<p>This file is identical to addemployee.cfm, which is called by StructNew, StructClear, and StructDelete. It adds employees. Employee information is passed through the employee structure (EMPINFO attribute). In UNIX, you must also add the Emp_ID.</p>

<!--- This view-only example shows the use of StructFind. --->

<cfswitch expression = "#ThisTag.ExecutionMode#" value = "start">
  <cfif StructIsEmpty(attributes.EMPINFO)>
    <cfoutput>Error. No employee data was passed.</cfoutput>
    <cfexit method = "ExitTag">
  <cfelse>
    <cfquery name = "AddEmployee" datasource = "cfdocexamples">
      INSERT INTO Employees (FirstName, LastName, Email, Phone, Department)
      VALUES
      ("#StructFind(attributes.EMPINFO, "firstname")#",
       "#StructFind(attributes.EMPINFO, "lastname")#",
       "#StructFind(attributes.EMPINFO, "email")#",
       "#StructFind(attributes.EMPINFO, "phone")#",
       "#StructFind(attributes.EMPINFO, "department")#")
    </cfquery>
  </cfelse>
  <cfoutput><hr>Employee Add Complete</cfoutput>
</cfcase>
</cfswitch> --->

© 2014 Adobe Systems Incorporated. All rights reserved.
**StructFindKey**

**Description**

Searches recursively through a substructure of nested arrays, structures, and other elements, for structures whose keys match the search key in the `value` parameter.

**Returns**

An array that contains structures with keys that match `value`.

**Category**

Structure functions

**Function syntax**

```
StructFindKey(top, value, scope)
```

**See also**

Structure functions; Structure functions in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>top</code></td>
<td>ColdFusion object (structure or array) from which to start search. This parameter requires an object, not a name of an object.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>String or a variable that contains one for which to search.</td>
</tr>
</tbody>
</table>
| `scope`   | • one: returns one matching key. Default.  
            • all: returns all matching keys.  
            If the key is not found, an empty array is returned. |

**Usage**

Returns an array that includes one structure for each of the specified keys it finds. The fields of each of these structures are:

- **Value**: value held in the found key
- **Path**: string that can be used to reach the found key
- **Owner**: parent object that contains the found key

A structure's keys are unordered.

**Example**
<cfset aResults = StructFindKey( cookie, "CFID" )>
StructFindValue

Description

Searches recursively through a substructure of nested arrays, structures, and other elements for structures with values that match the search key in the `value` parameter.

Returns

An array that contains structures keys whose values match the search key `value`. If none are found, returns an array of size 0.

Category

Structure functions

Function syntax

```coldfusion
StructFindValue( top, value [, scope])
```

See also

Structure functions; Structure functions in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>ColdFusion structure from which to start search. This parameter requires an object, not a name of an object.</td>
</tr>
<tr>
<td>value</td>
<td>String or a variable that contains one for which to search. The type must be a simple object. Arrays and structures are not supported.</td>
</tr>
</tbody>
</table>
| scope     | • one: function returns one matching key (default).  
            • all: function returns all matching keys. |

Usage

The fields of each structure in the returned array are:

- **Key**: name of the key in which the value was found
- **Path**: string which could be used to reach the found key
- **Owner**: parent object that contains the found key

A structure’s keys are unordered.

Example

```coldfusion
<cfset aResults = StructFindValue( cookie, "12345")>
```
StructGet

Description

Gets a structure(s) from a specified path.

Returns

An alias to the variable in the `pathDesired` parameter. If necessary, `StructGet` creates structures or arrays to make `pathDesired` a valid variable "path."

Category

Structure functions

Function syntax

```plaintext
StructGet (pathDesired)
```

See also

Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

History

ColdFusion MX:

- Changed behavior: this function can be used on XML objects.
- Changed behavior: if there is no structure or array present in `pathDesired`, this function creates structures or arrays to make `pathDesired` a valid variable "path."

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pathDesired</td>
<td>Pathname of variable that contains structure or array from which ColdFusion retrieves structure.</td>
</tr>
</tbody>
</table>

Usage

You can inadvertently create invalid structures using this function. For example, if array notation is used to expand an existing array, the specified new element is created, regardless of the type currently held in the array.

Example
<!--- GetStruct() test --->
<cfset test = StructGet( "dog.myscope.test" )>
<cfset test.foo = 1>
<cfif NOT IsDefined("dog")>
  Dog is not defined<br>
</cfif>
<cfif NOT IsDefined("dog.myscope")>
  Dog.Myscope is not defined<br>
</cfif>
<cfif NOT IsDefined("dog.myscope.test")>
  Dog.Myscope.Test is not defined<br>
</cfif>
<cfif NOT IsDefined("dog.myscope.test.foo")>
  Dog.Myscope.Test.Foo is not defined<br>
</cfif>
<cfoutput>
  #dog.myscope.test.foo#<br>
</cfoutput>
<cfset test = StructGet( "request.myscope[1].test" )>
<cfset test.foo = 2>
<cfoutput>
  #request.myscope[1].test.foo#<br>
</cfoutput>
<cfset test = StructGet( "request.myscope[1].test[2]" )>
<cfset test.foo = 3>
<cfoutput>
  #request.myscope[1].test[2].foo#<br>
</cfoutput>
**StructInsert**

**Description**

Inserts a key-value pair into a structure.

**Returns**

True, upon successful completion. If `structure` does not exist, or if `key` exists and `allowoverwrite = "False"`, ColdFusion throws an exception.

**Category**

Structure functions

**Function syntax**

```plaintext
StructInsert(structure, key, value [, allowoverwrite ])
```

**See also**

Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

**History**

ColdFusion MX: Changed behavior: this function can be used on XML objects.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure</td>
<td>Structure to contain the new key-value pair.</td>
</tr>
<tr>
<td>key</td>
<td>Key that contains the inserted value.</td>
</tr>
<tr>
<td>value</td>
<td>Value to add.</td>
</tr>
<tr>
<td>allowoverwrite</td>
<td>Optional. Whether to allow overwriting a key. The default value is False.</td>
</tr>
</tbody>
</table>

**Usage**

A structure’s keys are unordered.

**Example**
<h1>Add New Employees</h1>

<!--- Establish params for first time through --->
<cfparam name = "FORM.firstname" default = ""/>
<cfparam name = "FORM.lastname" default = ""/>
<cfparam name = "FORM.email" default = ""/>
<cfparam name = "FORM.phone" default = ""/>
<cfparam name = "FORM.department" default = ""/>

<cfif FORM.firstname EQ ">
<p>Please fill out the form.</p>
<cfelse>
<cfoutput>
<CFScript>
employee = StructNew();
StructInsert(employee, "firstname", FORM.firstname);
StructInsert(employee, "lastname", FORM.lastname);
StructInsert(employee, "email", FORM.email);
StructInsert(employee, "phone", FORM.phone);
StructInsert(employee, "department", FORM.department);
</CFScript>
<p>First name is #StructFind(employee, "firstname")#</p>
<p>Last name is #StructFind(employee, "lastname")#</p>
<p>EMail is #StructFind(employee, "email")#</p>
<p>Phone is #StructFind(employee, "phone")#</p>
<p>Department is #StructFind(employee, "department")#</p>
</cfoutput>

<!--- Call the custom tag that adds employees --->
<CF_ADDEMPLOYEE EMPINFO = "#employee#">
</cfif>

</cfoutput>

<Hr> <form action = "structinsert.cfm">
<p>First Name: &nbsp;
<input name = "firstname" type = "text" hspace = "30" maxlength = "30">
</p>
<p>Last Name: &nbsp;
<input name = "lastname" type = "text" hspace = "30" maxlength = "30">
</p>
<p>EMail: &nbsp;
<input name = "email" type = "text" hspace = "30" maxlength = "30">
</p>
<p>Phone: &nbsp;
<input name = "phone" type = "text" hspace = "20" maxlength = "20">
</p>
<p>Department: &nbsp;
<input name = "department" type = "text" hspace = "30" maxlength = "30">
</p>
<p>
<input type = "submit" value = "OK">
</form>
### StructIsEmpty

**Description**

Determines whether a structure contains data.

**Returns**

True, if structure is empty; if structure does not exist, ColdFusion throws an exception.

**Category**

[Decision functions](#), [Structure functions](#)

**Function syntax**

```
StructIsEmpty(structure)
```

**See also**

[Structure functions](#); [Modifying a ColdFusion XML object](#) in the Developing *ColdFusion Applications*

**History**

ColdFusion MX: Changed behavior: this function can be used on XML objects.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure</td>
<td>Structure to test</td>
</tr>
</tbody>
</table>

**Example**
This example illustrates use of StructIsEmpty. This file is identical to addemployee.cfm, which is called by StructNew, StructClear, and StructDelete. It adds employees. Employee information is passed through employee structure (EMPINFO attribute). In UNIX, you must also add the Emp_ID.

```coldfusion
<cfswitch expression = "#ThisTag.ExecutionMode#"
<cfcase value = "start">
  <cfif StructIsEmpty(attributes.EMPINFO)>
    <cfoutput>Error. No employee data was passed.</cfoutput>
    <cfexit method = "ExitTag">
  <cfelse>
    <!--- Add the employee; In UNIX, you must also add the Emp_ID --->
    <cfquery name = "AddEmployee" datasource = "cfdocexamples">
      INSERT INTO Employees (FirstName, LastName, Email, Phone, Department) VALUES
      <cfoutput>
        ('#StructFind(attributes.EMPINFO, "firstname")#',
        '#StructFind(attributes.EMPINFO, "lastname")#',
        '#StructFind(attributes.EMPINFO, "email")#',
        '#StructFind(attributes.EMPINFO, "phone")#',
        '#StructFind(attributes.EMPINFO, "department")#'
      </cfoutput>
    </cfquery>
    <cfoutput><hr>Employee Add Complete</cfoutput>
  </cfelse>
</cfswitch>
```
**StructKeyArray**

**Description**

Finds the keys in a ColdFusion structure.

**Returns**

An array of keys; if `structure` does not exist, ColdFusion throws an exception.

**Category**

Structure functions

**Function syntax**

```
StructKeyArray(structure)
```

**See also**

Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure</td>
<td>Structure from which to extract a list of keys</td>
</tr>
</tbody>
</table>

**Usage**

A structure’s keys are unordered.

**Example**

```html
<!--- Shows StructKeyArray function to copy keys from a structure to an array. Uses StructNew to create structure and fills its fields with the information the user enters in the form fields. --->
<h3>StructKeyArray Example</h3>

<h3>Extracting the Keys from the Employee Structure</h3>

<!-- Create structure. Check whether Submit was pressed. If so, define fields in employee structure with user entries on form. -->
<cfset employee = StructNew()>
<cfif Isdefined("Form.Submit")>
  <cfif Form.Submit is "OK">
    <cfset employee.firstname = FORM.firstname>
    <cfset employee.lastname = FORM.lastname>
    <cfset employee.email = FORM.email>
    <cfset employee.phone = FORM.phone>
    <cfset employee.company = FORM.company>
  </cfelseIf Form.Submit is "Clear">
  <cfset rc = StructClear(employee)>
</cfif>
</cfif>
```
This example uses the StructNew function to create a structure called "employee" that supplies employee info. Its fields are filled by the form. After you enter employee information in structure, the example uses StructKeyArray function to copy all of the keys from the structure into an array. </p>
<hr size = "2" color = "#0000A0">
<form action = "structkeyarray.cfm">
<table cellspacing = "2" cellpadding = "2" border = "0">
<tr>
<td>First Name:</td>
<td><input name = "firstname" type = "text"
    value = "" hspace = "30" maxlength = "30"></td>
</tr>
<tr>
<td>Last Name:</td>
<td><input name = "lastname" type = "text"
    value = "" hspace = "30" maxlength = "30"></td>
</tr>
<tr>
<td>EMail</td>
<td><input name = "email" type = "text"
    value = "" hspace = "30" maxlength = "30"></td>
</tr>
<tr>
<td>Phone:</td>
<td><input name = "phone" type = "text"
    value = "" hspace = "20" maxlength = "20"></td>
</tr>
<tr>
<td>Company:</td>
<td><input name = "company" type = "text"
    value = "" hspace = "30" maxlength = "30"></td>
</tr>
<tr>
<td><input type = "submit" name = "submit"
    value = "OK"></td>
<td><b>After you submit the FORM, scroll down to see the array.</b></td>
</tr>
</table>
</form>
<cfif NOT StructISEmpty(employee)>
<hr size = "2" color = "#0000A0">
<cfset keysToStruct = StructKeyArray(employee)>
<cfloop index = "i" from = "1" to = "#ArrayLen(keysToStruct)#">
<p><cfoutput>Key#i# is #keysToStruct[i]#</cfoutput></p>
<p><cfoutput>Value#i# is #employee[keysToStruct[i]]#</cfoutput></p>
</cfloop>
</cfif>
</cfif>
</cfloop>
**StructKeyExists**

**Description**

Determines whether a specific key is present in a structure.

**Returns**

True, if *key* is in *structure*.

**Category**

Decision functions, Structure functions

**Function syntax**

```
StructKeyExists(structure, "key")
```

**See also**

Structure functions: Structure functions in the *Developing ColdFusion Applications*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure</td>
<td>Name of structure to test</td>
</tr>
<tr>
<td>key</td>
<td>Key to test</td>
</tr>
</tbody>
</table>

**Usage**

This function can sometimes be used in place of the `IsDefined` function, when working with the URL and Form scopes, which are structures. The following pieces of code are equivalent:

```
<cfif IsDefined("Form.JediMaster")>
<cfif StructKeyExists(Form,"JediMaster")>
```

A structure’s keys are unordered.

**Example**
<!--- This example shows the use of StructKeyExists. --->
<p>This file is similar to addemployee.cfm, which is called by StructNew, StructClear, and StructDelete. To test, copy the &lt;LT;CFELSEif&gt; statement to the appropriate place in addemployee.cfm. It is a custom tag to add employees. Employee information is passed through the employee structure (the EMPINFO attribute). In UNIX, you must also add the Emp_ID.</p>

<cfswitch expression = "#ThisTag.ExecutionMode#">
  <cfcase value = "start">
    <cfif StructIsEmpty(attributes.EMPINFO)>
      <cfoutput>Error. No employee data was passed.</cfoutput>
      <cfexit method = "ExitTag">
    </cfif>
    <cfelseif NOT StructKeyExists(attributes.EMPINFO, "department")>
      <cfscript>
        StructUpdate(attributes.EMPINFO, "department", "Unassigned");
      </cfscript>
      <cfexit method = "ExitTag">
    </cfelse>
  </cfcase>
  <cfelse>
    <cfexit method = "ExitTag">
    </cfelse>
</cfswitch>
StructKeyList

Description

Extracts keys from a ColdFusion structure.

Returns

A list of keys; if structure does not exist, ColdFusion throws an exception.

Category

Structure functions

Function syntax

StructKeyList(structure [, delimiter])

See also

Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure</td>
<td>Structure from which to extract a list of keys.</td>
</tr>
<tr>
<td>delimiter</td>
<td>Optional. Character that separates keys in list. The default value is comma.</td>
</tr>
</tbody>
</table>

Usage

A structure's keys are unordered.

Example

```coldfusion
<!--- This example shows how to use StructKeyList to list the keys in a structure. It uses StructNew function to create structure and fills it with information user enters in form fields. --->

<!--- This section creates structure and checks whether Submit has been pressed. If so, code defines fields in the employee structure with what the user entered in the form. --->
<cfset employee = StructNew()>
<cfif IsDefined("Form.Submit")>
  <cfif Form.Submit is "OK">
    <cfset employee.firstname = FORM.firstname>
    <cfset employee.lastname = FORM.lastname>
    <cfset employee.email = FORM.email>
    <cfset employee.phone = FORM.phone>
    <cfset employee.company = FORM.company>
  </cfelseIf Form.Submit is "Clear">
  <cfset rc = StructClear(employee)>
```
<h3>StructKeyList Function</h3>
<h3>Listing the Keys in the Employees Structure</h3>
<p>This example uses StructNew function to create structure "employee" that supplies employee information. The fields are filled with the contents of the following form.</p>
<p>After you enter employee information into structure, example uses StructKeyList() function to list keys in structure.</p>
<p>This code does not show how to insert information into a database. See cfquery for more information about database insertion.</p>
<hr size = "2" color = "#0000A0">
<form action = "structkeylist.cfm">
<table cellspacing = "2" cellpadding = "2" border = "0">
<tr><td>First Name:</td><td><input name = "firstname" type = "text" value = "" hspace = "30" maxlength = "30"></td></tr>
<tr><td>Last Name:</td><td><input name = "lastname" type = "text" value = "" hspace = "30" maxlength = "30"></td></tr>
<tr><td>Email</td><td><input name = "email" type = "text" value = "" hspace = "30" maxlength = "30"></td></tr>
<tr><td>Phone:</td><td><input name = "phone" type = "text" value = "" hspace = "20" maxlength = "20"></td></tr>
<tr><td>Company:</td><td><input name = "company" type = "text" value = "" hspace = "30" maxlength = "30"></td></tr>
<tr><td><input type = "submit" name = "submit" value = "OK"></td><td><b>After you submit form, scroll down to see the list.</b></td></tr>
</table>
</form>
<cfif NOT StructISEmpty(employee)>
<hr size = "2" color = "#0000A0">
<cfset keysToStruct = StructKeyList(employee,"<li>")>
<p>Here are the keys to the structure:</p>
<ul><li><cfoutput>#keysToStruct#</cfoutput></li></ul>
<p>If fields are correct, we can process new employee information.</p>
If they are not correct, consider rewriting application.</p>
</cfif>
**StructNew**

Description

Creates a structure.

Returns

A structure.

Category

Structure functions

Function syntax

```
StructNew()
```

See also

Structure functions; Structure functions in the Developing ColdFusion Applications

Parameters

None

Example
<h1>Add New Employees</h1>

<cfparam name = "FORM.firstname" default = "">
<cfparam name = "FORM.lastname" default = "">
<cfparam name = "FORM.email" default = "">
<cfparam name = "FORM.phone" default = "">
<cfparam name = "FORM.department" default = "">
<cfif FORM.firstname EQ "">
<p>Please fill out the form.</p>
<cfelse>
<cfoutput>
<cfscript>
employee = StructNew();
StructInsert(employee, "firstname", FORM.firstname);
StructInsert(employee, "lastname", FORM.lastname);
StructInsert(employee, "email", FORM.email);
StructInsert(employee, "phone", FORM.phone);
StructInsert(employee, "department", FORM.department);
</cfscript>
<p>First name is #StructFind(employee, "firstname")#</p>
<p>Last name is #StructFind(employee, "lastname")#</p>
<p>Email is #StructFind(employee, "email")#</p>
<p>Phone is #StructFind(employee, "phone")#</p>
<p>Department is #StructFind(employee, "department")#</p>
</cfoutput>
</cfif>

<!--- Call the custom tag that adds employees --->
<CF_ADDEMPLOYEE EMPINFO = "#employee#">
</cfif>
StructSort

Description

Returns a sorted array of the top level keys in a structure. Sorts using alphabetic or numeric sorting, and can sort based on the values of any structure element.

Returns

An array of top-level key names (strings), sorted by the value of the specified subelement.

Category

Structure functions

Function syntax

```
StructSort(base[, sortType[, sortOrder[, pathToSubElement[, localeSensitive]]]])
```

See also

Structure functions; Structure functions in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>A ColdFusion structure</td>
</tr>
</tbody>
</table>
| sortType              | • numeric  
• text: case sensitive (all lowercase letters precede the first uppercase letter). Default.  
• textnocase                                                                                                                                 |
| sortOrder             | • asc: ascending (a to z) sort order. Default.  
• desc: descending (z to a) sort order                                                                                                                                 |
| pathToSubElement      | String or a variable that contains one.Path to apply to each top-level key, to reach element value by which to sort. The default value is nothing (top-level entries sorted by their own values). |
| localeSensitive       | Specify if you wish to do a locale sensitive sorting. The default value is false.                                                           |

Usage

The pathToSubElement string does not support array notation, and only supports substructures of structures. This function does not sort or change the structure. In ColdFusion 10, added support for all Java supported locale-specific characters (including support for umlaut characters). A flag for this support has been added for sorttype = "text" or sorttype = "textnocase".
Example

```<cfscript>
salaries = StructNew() ;
employees = StructNew() ;
departments = StructNew() ;
for ( i=1; i lt 6; i=i+1 )
{
    salary = 120000 - i*10000 ;
salaries["employee#i#"] = salary ;

    employee = StructNew() ;
    employee["salary"] = salary ;
    // employee.salary = salary ;
    employees["employee#i#"] = employee ;

    departments["department#i#"] = StructNew() ;
departments["department#i#"].boss = employee ;
}
</cfscript>
<cfoutput>
<p>list of employees based on the salary (text search): <br>
1) #ArrayToList( StructSort( salaries ) )#<br>
2) #ArrayToList( StructSort( salaries, "text", "ASC" ) )#<br>
3) #ArrayToList( StructSort( salaries, "textnocase", "ASC" ) )#<br>
4) #ArrayToList( StructSort( salaries, "text", "DESC" ) )#<br>
<p>list of employees based on the salary (numeric search): <br>
5) #ArrayToList( StructSort( salaries, "numeric", "ASC" ) )#<br>
6) #ArrayToList( StructSort( salaries, "numeric", "DESC" ) )#<br>
<p>list of employees based on the salary (subfield search): <br>
7) #ArrayToList( StructSort( employees, "numeric", "DESC", "salary" ) )#<br>
8) #ArrayToList( StructSort( employees, "text", "ASC","salary" ) )#<br>
<p>list of departments based on the salary (sub-sub-field search): <br>
9) #ArrayToList( StructSort( departments, "text", "ASC", "boss.salary" ) )#<br>
</cfoutput>

<!---- add an invalid element and test that it throws an error --->
<p><p>
<cfset employees[ "employee4" ] = StructNew()>
<cftry>
<cfset temp = StructSort( employees, "text", "ASC", "salary" )>
<cfoutput>We have a problem - this was supposed to throw an exception!</br>
</cfoutput>
<cfcatch type="any">
<cfoutput>
ERROR: <b>This error was expected!</b><br>
#cfcatch.message# - #cfcatch.detail#<br>
</cfoutput>
</cfcatch>
</cftry>
```

**StructUpdate**

**Description**

Updates a key with a value.

**Returns**

True, on successful execution; if the structure does not exist, ColdFusion throws an error.

**Category**

Structure functions

**Function syntax**

```
StructUpdate(structure, key, value)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structure</td>
<td>Structure to update</td>
</tr>
<tr>
<td>key</td>
<td>Key, the value of which to update</td>
</tr>
<tr>
<td>value</td>
<td>New value</td>
</tr>
</tbody>
</table>

**See also**

Structure functions; Modifying a ColdFusion XML object in the Developing ColdFusion Applications

**History**

ColdFusion MX: Changed behavior: this function can be used on XML objects.

**Example**
This example shows the use of StructUpdate. This file is similar to addemployee.cfm, which is called by StructNew, StructClear, and StructDelete. To test this file, copy the &LT;CFELSEIF&GT; statement to the appropriate place in addemployee.cfm. It is an example of a custom tag used to add employees. Employee information is passed through the employee structure (the EMPINFO attribute). In UNIX, you must also add the Emp_ID.

```cfml
<cfswitch expression = "#ThisTag.ExecutionMode#">
  <cfcase value = "start">
    <cfif StructIsEmpty(attributes.EMPINFO)>
      <cfoutput>Error. No employee data was passed.</cfoutput>
      <cfexit method = "ExitTag">
    </cfelseIf
    <cfscript>
      StructUpdate(attributes.EMPINFO, "department", "Unassigned");
    </cfscript>
    <cfexit method = "ExitTag">
  </cfelse>
  #back to top
```

SpreadSheetAddPagebreaks

Description

A function to add page breaks for rows and columns.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

```
SpreadSheetAddPagebreaks(Spreadsheetobj, rowbreaks, colbreaks )
```

History

ColdFusion 11: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadSheetObj</td>
<td>The Excel spreadsheet object to which to add the column.</td>
</tr>
<tr>
<td>rowbreaks</td>
<td>Comma-delimited row numbers where the page breaks will be applied.</td>
</tr>
<tr>
<td>colbreaks</td>
<td>Comma-delimited column numbers where the page breaks will be applied.</td>
</tr>
</tbody>
</table>

Example
<cfset rowCount = 200>
<cfset colCount = 150>

<cfloop from="1" to="#colCount#" index="i">
  <cfset val_arr[i] = "some random text row#i#">
</cfloop>

<cfset val_list = #ArrayToList(val_arr)#>
<cfset x1Obj = spreadsheetNew("testsheet", true)>

<cfloop from="1" to="#rowCount#" index="r">
  <cfset spreadsheetAddRow(x1Obj, "#val_list#")>
</cfloop>

<cfset SpreadSheetAddPagebreaks(x1Obj, "50,90", "4,9")>
<cfset SpreadSheetWrite(x1Obj, "#Expandpath("./")#test_pagebreak.xlsx",true)>
SpreadSheetAddAutofilter

Description

A function to add auto filters to the spreadsheet.

Returns

Does not return a value.

Category

Microsoft Office Integration

Function syntax

SpreadSheetAddAutofilter(Spreadsheetobj, autofilter )

History

ColdFusion 11: Added the function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spreadsheetObj</td>
<td>The Excel spreadsheet object to which to add the column.</td>
</tr>
<tr>
<td>autofilter</td>
<td>The Excel filter that needs to be applied to the sheet. Note that the value should not contain spaces or invalid characters.</td>
</tr>
</tbody>
</table>

Example
<cfset colList = "col1,col2,col3,col4,col5,col6,col7,col8,co9,co10">
<cfset rowCount = 100>
<cfset qryObj = QueryNew("#colList#")>
<cfset QueryAddRow(qryObj, #rowCount#)>

<cfloop from="1" to="#rowCount#" index="r">
  <cfloop from="1" to="#ListLen(colList)#" index="c">
    <cfset QuerySetCell(qryObj, #ListGetAt(colList,c)#, "some random text
c", r)
  </cfloop>
</cfloop>

<cfset xlObj = spreadsheetNew("testsheet", true)>
<cfset spreadsheetAddRows(xlObj, #qryObj#)>
<cfset SpreadSheetAddAutofilter(xlObj, "E5:J50")>
<cfset spreadsheetwrite(xlObj, #Expandpath("./")#test_autofilter.xlsx", true)>
StructReduce

Description

Iterates over every entry of the struct and calls the closure to work on the key value pair of the struct. This function will reduce the struct to a single value and will return the value.

Returns

any

Syntax

```
StructReduce(struct, function(result, key, value [,struct]), initialVal)
```

History

ColdFusion 11: Added this function.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct</td>
<td>Required</td>
<td></td>
<td>The input struct.</td>
</tr>
<tr>
<td>function</td>
<td>Required</td>
<td></td>
<td>Closure or a function reference that will be called for each of the iteration. The arguments passed to the callback are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• result: the result of the reduce operation after the previous iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• key: key for the current iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• value: the value for the current iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• struct: a reference of the original struct</td>
</tr>
<tr>
<td>initialVal</td>
<td>Optional</td>
<td></td>
<td>Initial value which will be used for the reduce operation. The type is any.</td>
</tr>
</tbody>
</table>

Example
<cfscript>

person = {fname="John", lname="Doe");

ucaseStruct = structMap(person, function(key, value)
{
    return ucase(value);
});

writeDump(ucaseStruct);

concatValues = structReduce(person, function(result, key, value)
{
    result = result?:"";
    result&= value & " ";
    return result;
});

writeDump(concatValues);

</cfscript>
StructMap

Description

Iterates over every entry of the Struct and calls the closure function to work on the key value pair of the struct. The returned value will be set for the same key in a new struct and the new struct will be returned.

Returns

Struct

Syntax

```
StructMap(struct, function(key, value [,struct]))
```

History

ColdFusion 11: Added this function.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct</td>
<td>Required</td>
<td></td>
<td>The input struct.</td>
</tr>
<tr>
<td>function</td>
<td>Required</td>
<td></td>
<td>Closure or a function reference that will be called for each of the iteration. The arguments passed to the callback are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• key: key for the current iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• value: the value for the current iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• struct: a reference of the original struct</td>
</tr>
</tbody>
</table>
Serialize

Description

Serializes the object to a specified type

Returns

String

Function syntax

```
serialize( Object objToBeSerialized, String type, boolean useCustomSerializer )
```

History

ColdFusion 11: Added this function. For more information see REST Enhancements in ColdFusion 11.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objToBeSerialized</td>
<td>Required. An object to be serialized.</td>
</tr>
<tr>
<td>type</td>
<td>A type to which the object will be serialized. ColdFusion, by default supports XML and JSON formats. You can also implement support for other types in the CustomSerializer CFC.</td>
</tr>
<tr>
<td>useCustomSerializer</td>
<td>Boolean. Whether to use the custom serializer or not. The default value is true. The custom serializer will be always used for XML deserialization. If false, the XML/JSON deserialization will be done using the default ColdFusion behavior. If any other type is passed with useCustomSerializer as false, then TypeNotSupportedException will be thrown.</td>
</tr>
</tbody>
</table>

For more information on usage, see this document.
SerializeXML

Description

Serializes a ColdFusion object to XML.

Returns

String

Function syntax

```
serializeXml( Object objToBeSerialized, boolean useCustomSerializer )
```

History

ColdFusion 11: Added this function. For more information see REST Enhancements in ColdFusion 11.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objToBeSerialized</td>
<td>Required. An object to be serialized.</td>
</tr>
<tr>
<td>useCustomSerializer</td>
<td>Boolean. Whether to use the custom serializer or not. The default value is true. The custom serializer will be always used for XML deserialization. If false, the XML/JSON deserialization will be done using the default ColdFusion behavior. If any other type is passed with useCustomSerializer as false, then TypeNotSupportedException will be thrown.</td>
</tr>
</tbody>
</table>

For more information on usage, see this document.
Functions t-z

- Tan
- ThreadJoin
- ThreadTerminate
- Throw
- TimeFormat
- ToBase64
- ToBinary
- ToScript
- ToString
- Trace
- Transactionandconcurrency
- TransactionCommit
- TransactionRollback
- TransactionSetSavePoint
- Trim
- UCase
- URL Decode
- URL EncodedFormat
- URLSessionFormat
- Val
- ValueList
- VerifyClient
- Week
- Wrap
- WriteDump
- WriteLog
- WriteOutput
- WSGlobalChannels
- WSGlobalSubscribers
- WSPublish
- WSSendMessage
- XmlChildPos
- XmlElemNew
- XmlFormat
- XmlNodeType
- XmlNew
- XmlParse
- XmlSearch
- XmlTransform
- XmlValidate
- Year
- YesNoFormat
Tan

Description

Calculates the tangent of an angle that is entered in radians.

Returns

A number; the tangent of an angle.

Category

Mathematical functions

Function syntax

```
Tan(number)
```

See also

Atn, Cos, ACos, Sin, ASin, Pi

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Angle, in radians, for which to calculate the tangent.</td>
</tr>
</tbody>
</table>

Usage

To convert degrees to radians, multiply degrees by p/180. To convert radians to degrees, multiply radians by 180/p.

⚠️ Note

Because the function uses floating point arithmetic, it can return a small number (such as 6.1232399574E-017) for angles that must produce 0 and can return a large number (such as 1.63312393532E+016) for infinity or not a number. To test for a 0 value, check whether the value is less than 0.0000000000001. To test for an infinite value, check whether the value is more than 1E15.

Example
<h3>Tan Example</h3>

<!--- Calculate tangent if form has been submitted --->
<cfif IsDefined("FORM.tanNum")>

<!--- Make sure input is a number --->
<cfif IsNumeric(#FORM.tanNum#)>

<!--- Convert degrees to radians, call the Tan function. --->
<cfset tanValue=#Tan((Form.tanNum * PI()) / 180)#>

<!--- 0.0000000000001 is the function's precision limit. If absolute value of returned value is less, set result to 0 --->
<cfif Abs(tanValue) LT 0.0000000000001>
<cfset tanValue=0>
</cfif>

<cfoutput>
Tan(#FORM.tanNum#) = #tanValue#<br><br>
</cfoutput>

<cfelse>

<!--- If input is not a number, show an error message --->
<h4>You must enter a numeric angle in degrees.</h4>
</cfif>

</cfif>

<form action = "#CGI.script_name#" method="post">
Enter an angle in degrees to get its tangent:
<br><input type = "Text" name = "tanNum" size = "15">
<br><br>
<input type = "Submit" name = "">&nbsp;&nbsp;
<input type = "RESET">
</form>
ThreadJoin

Description

Makes the current thread wait until the thread or threads specified complete processing.

Category

Exception handling functions, Data output functions

Function syntax

```plaintext
ThreadJoin([threadName], [timeout])
```

See also

cfscript, cfthrow, cftry, cfcatch

History

ColdFusion 9: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>threadName</td>
<td>The name of the thread or threads to join to the current thread. To specify multiple threads, use a comma-separated list.</td>
</tr>
<tr>
<td>timeout</td>
<td>The number of milliseconds for which to suspend thread processing.</td>
</tr>
</tbody>
</table>

Usage

Makes the current thread wait until the thread or threads specified in the (`threadName`) parameter complete processing, or until the period specified in the (`timeout`) parameter passes, before continuing processing.

- ThreadJoin(): Current thread waits for all ColdFusion threads to complete processing.
- ThreadJoin(`threadName`): Makes current thread wait for the specified thread to finish execution.
- ThreadJoin(`threadName, timeout`): Makes the current thread wait until execution timeout for one or many threads specified by `threadName`. If you do not specify a timeout and the thread you are joining to does not finish, the current thread also cannot finish processing.

Example
<cfscript>
thread name="t1"
{
    sleep(5000);
}
thread name="t2"
{
    threadjoin("t1",1000);
}
threadjoin("t2");
</cfscript>
<cfoutput>Status of the thread T1 = #t1.Status#<br></cfoutput>
<cfoutput>Status of the thread T2 = #t2.Status#<br></cfoutput>
ThreadTerminate

Description

Terminates the thread specified by threadName. Behaves same as cfthread action="terminate".

Category

Exception handling functions, Data output functions

Function syntax

ThreadTerminate(threadName)

See also

cfscript, cfthrow, cftry, cfcatch

History

ColdFusion 9: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>threadName</td>
<td>The name of the thread to stop.</td>
</tr>
</tbody>
</table>

Usage

Use this function to stop processing of the thread specified in the threadName. If you terminate a thread, the thread scope includes an ERROR metadata structure that provides information about the termination.

Example

```
<cfscript>
    thread name="t1"
    {
        sleep(3000);
    }
    sleep(1000);
    threadTerminate("t1");
    sleep(1000);
</cfscript>
<coutput>Status of the thread = #t1.Status#</coutput>
```
Throw

Description

A function equivalent of the `cfthrow` tag and is used in the `<cfscript>` mode.

Parameters

Same as the `<cfthrow>` tag.

Category

Exception handling functions, Data output functions

Function syntax

For name=value pair:

```
throw (message = "message", type = "exception type", detail, errorCode = "error code", extendedInfo = "additional info", object = "java exception object")
```

For positional notations, the sequence must be followed exactly in the same manner as provided in the syntax. If you do not provide one of the attributes, use an empty string instead.

History

ColdFusion 11: Attributes passed to this function needs to be in a comma-separated format.

ColdFusion 9: Added this function.

See also

cfscript, cfthrow, cftry, cfcatch

Usage

You can call this function by passing arguments as name=value pair or as positional arguments. For positional notations, specify the arguments in the sequence mentioned in the function syntax.

Example
<cfscript>
function TotalInterest(principal, annualRate, months)
{
    var years = 0;
    var interestRate = 0;
    var totalInterest = 0;
    principal = REReplace(trim(principal), "\[\$\]", ",", "ALL");
    annualRate = Replace(trim(annualRate), ",\%", ",", "ALL");

    if ((principal <= 0) OR (annualRate <= 0) OR (months <= 0)) {
        Throw(type="InvalidData",message="All values must be greater than 0.");

        //Use of Throw function in cfscript
    }
    interestRate = annualRate / 100;
    years = months / 12;
    totalInterest = principal * (((1 + interestRate) ^ years) - 1);
    return DollarFormat(totalInterest);
}

try {
    WriteOutput(TotalInterest("$2500.00", "5.5\%", "12");
} catch(InvalidData ex) {
    //Displaying exception details on screen
    WriteOutput("<p>An InvalidData exception was thrown.</p>"=
    WriteOutput("<p>#ex.message#</p>");
}
</cfscript>

From ColdFusion 11, attributes passed to this function needs to be in a comma-separated format.
**TimeFormat**

**Description**

Formats a time value using U.S. English time formatting conventions.

**Returns**

A custom-formatted time value. If no mask is specified, returns a time value using the \texttt{hh:nn tt} format. For international time formatting, see \texttt{LSTimeFormat}.

**Category**

\textbf{Date and time functions, Display and formatting functions}

**Function syntax**

\begin{verbatim}
TimeFormat(time [, mask ])
\end{verbatim}

**See also**

\texttt{CreateTime, Now, ParseDateTime, LSTimeFormat, DateFormat}

**History**

ColdFusion 10: The mask "m or M" for minute is deprecated. Recommended "n or N".

ColdFusion MX 6.1: Added the mask character L or l to represent milliseconds.

ColdFusion MX:

- Changed the way extra characters are processed: this function processes extra characters within the mask value differently than in earlier releases, as follows:
  - ColdFusion 5 and earlier: the function returns the time format and an apostrophe-delimited list of the extra characters. For example, \texttt{TimeFormat(Now(), "hh:mm:ss dog")} returns \texttt{8:17:23 d'o'g}.
  - ColdFusion MX: the function returns the time format and the extra characters. For example, for the call above, it returns \texttt{8:17:23 dog}.
    If the extra characters are single-quoted (for example, \texttt{hh:mm:ss 'dog'}), ColdFusion 5 and ColdFusion MX return the time format and the extra characters: \texttt{8:17:23 dog}.
- Added support for the following mask parameter options: short, medium, long, and full.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>A date/time value or string to convert</td>
</tr>
</tbody>
</table>
Masking characters that determine the format:

- **h**: hours; no leading zero for single-digit hours (12-hour clock)
- **hh**: hours; leading zero for single-digit hours (12-hour clock)
- **H**: hours; no leading zero for single-digit hours (24-hour clock)
- **HH**: hours; leading zero for single-digit hours (24-hour clock)
- **n**: minutes; no leading zero for single-digit minutes
- **nn**: minutes; a leading zero for single-digit minutes
- **s**: seconds; no leading zero for single-digit seconds
- **ss**: seconds; leading zero for single-digit seconds
- **l** or **L**: milliseconds, with no leading zeros
- **t**: one-character time marker string, such as A or P
- **tt**: multiple-character time marker string, such as AM or PM
- **short**: equivalent to h:mm tt
- **medium**: equivalent to h:mm:ss tt
- **long**: medium followed by three-letter time zone; as in, 2:34:55 PM EST
- **full**: same as long

### Usage

When passing a date/time value as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date/time object. Database query results for date and time values can vary in sequence and formatting unless you use functions to format the results. To ensure that dates and times display with appropriate formatting, and that users of your ColdFusion application are not confused by dates and times displayed, Adobe recommends that you use the **DateFormat** and **TimeFormat** functions to format date and time values from queries. For more information and examples, see TechNote, "*ColdFusion (5 and 4.5.x) with Oracle: Formatting Date and Time Query Results,*" at [go.adobe.com/kb/ts_tn_18070_en-us](go.adobe.com/kb/ts_tn_18070_en-us).

### Example
<cfset todayDate = #Now()#>
<body>
<h3>TimeFormat Example</h3>
<p>Today's date is <cfoutput>#todayDate#</cfoutput>.</p>
<p>Using Timeformat, we can display the value in different ways:</p>
<cfoutput>
<ul>
<li>#TimeFormat(todayDate)#</li>
<li>#TimeFormat(todayDate, "hh:nn:ss")#</li>
<li>#TimeFormat(todayDate, "hh:nn:ssst")#</li>
<li>#TimeFormat(todayDate, "hh:nn:ssstt")#</li>
<li>#TimeFormat(todayDate, "HH:nn:ss")#</li>
</ul>
</cfoutput>
<p>To generate a standard ISO 8601 W3C Date and Time string like 1997-07-16T19:20, concatenate a DateFormat function, the character T, and a TimeFormat function. For example: dateformat(now(), "yyyy-mm-dd")#T#TimeFormat(now(), "HH:nn:ss") produces:</p>
<cfoutput>#dateformat(now(), "yyyy-mm-dd")#T#TimeFormat(now(), "HH:nn:ss")#</cfoutput>
</body>
ToBase64

Description

Calculates the Base64 representation of a string or binary object. The Base64 format uses printable characters, allowing binary data to be sent in forms and e-mail, and stored in a database or file.

Returns

The Base64 representation of a string or binary object.

Category

Conversion functions, String functions

Function syntax

ToBase64(string or binary_object [, encoding])

See also

- BinaryEncode for conversion of binary data to base64
- cffile for information about loading and reading binary data
- cfwddx for information about serializing and deserializing binary data
- IsBinary and ToBinary for checking for binary data and converting a Base64 object to binary format

History

ColdFusion MX: Added the encoding parameter.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string or binary_object</td>
<td>A string, the name of a string, or a binary object.</td>
</tr>
</tbody>
</table>
For a string, defines how characters are represented in a byte array. The following list includes commonly used values:

- utf-8
- iso-8859-1
- windows-1252
- us-ascii
- shift_jis
- iso-2022-jp
- euc-jp
- euc-kr
- big5
- euc-cn
- utf-16

For more information on character encoding, see: [www.w3.org/International/O-charset.html](http://www.w3.org/International/O-charset.html). The default value is the encoding of the page on which the function is called. See `cfcontent`. For a binary object, this parameter is ignored.

Usage

Adobe recommends that you use the `BinaryEncode` function to convert binary data to Base64-encoded data in all new applications.

```cfc
<h3>ToBase64 Example</h3>
<!--- Initialize data. ---->
<cfset charData = ""

<!--- Create string of ASCII characters (32-255); concatenate them --->
<cfloop index = "data" from = "32" to = "255">
  <cfset ch = chr(data)>
  <cfset charData = charData & ch>
</cfloop>
<p>
The following string is the concatenation of all characters (32 to 255) from the ASCII table.<br>
#charData#</p>

<!--- Create a Base64 representation of this string. ---->
<cfset data64 = toBase64(charData)>

<!--- Convert string to binary. -------->
<cfset binaryData = toBinary(data64)>

<!--- Convert binary back to Base64. ---->
<cfset another64 = toBase64(binaryData)>

<!--- Compare another64 with data64 to ensure that they are equal. ---->
<cfif another64 eq data64>
  <h3>Base64 representations are identical.</h3>
</cfif>
<cfelse>
  <h3>Conversion error.</h3>
</cfelse>
</cfif>
```
ToBinary

Adobe recommends that you use the **BinaryDecode** function to convert Base64 encoded data to binary data in all new applications.

**Description**

Calculates the binary representation of Base64-encoded data, PDF document, or a spreadsheet.

**Returns**

A binary representation of the data.

**Category**

Conversion functions, String functions

**Function syntax**

```
ToBinary(Data)
```

**See also**

- **BinaryDecode** for conversion of binary-encoded data, including Base64, to binary data
- **cffile** for information about loading and reading binary data
- **{cfwddx}** for information about serializing and deserializing binary data
- **IsBinary** and **ToBase64** for checking format and converting to Base64
- **Len** for determining the length of a binary object
- **Binary data type and binary encoding** in [Data types- Developing guide](#) in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>A variable representing data in Base64-encoded format or a PDF document.</td>
</tr>
</tbody>
</table>

**Usage**

The **ToBinary** function can take as a parameter a PDF document variable (specified by the **cfpdf** tag name attribute). In this case, the **ToBinary** function returns a byte array (byte[]) representation of the document. You can use the results of this function, for example, to store the PDF in a database as a BLOB, or, in a **cfcontent** tag, to write the PDF to the browser. You can use this binary representation with a read operation in the **cfpdf** tag to create a variable. The following example reads an unprotected PDF file, applies protections, and displays it in the browser:
Adobe recommends that you use the `BinaryDecode` function to convert Base64 encoded data to binary data in all new applications. If you pass a binary value to this function, it returns the input value.

Example

```cfml
<h3>ToBinary Example</h3>
<!---- Initialize data. ---->
<cfset charData = ">
<!---- Create a string of ASCII characters (32-255); concatenate them. ---->
<cfloop index = "data" from = "32" to = "255">
  <cfset ch = chr(data)>
  <cfset charData = charData & ch>
</cfloop>
<p>The following string is the concatenation of all characters (32 to 255) from the ASCII table.<br>
<cfoutput>#charData#</cfoutput></p>
<!---- Create a Base64 representation of this string. ---->
<cfset data64 = toBase64(charData)>

<!---- Convert string to binary. ---->
<cfset binaryData = toBinary(data64)>
<!---- Convert binary back to Base64. ---->
<cfset another64 = toBase64(binaryData)>
<!---- Compare another64 with data64 to ensure that they are equal. ---->
<cfif another64 eq data64>
  <h3>Base64 representation of binary data is identical to the Base64 representation of string data.</h3>
</cfif>
<h3>Conversion error.</h3>
</cfif>
```
ToScript

Description

Creates a JavaScript or ActionScript expression that assigns the value of a ColdFusion variable to a JavaScript or ActionScript variable. This function can convert ColdFusion strings, numbers, arrays, structures, and queries to JavaScript or ActionScript syntax that defines equivalent variables and values.

Returns

A string that contains a JavaScript or ActionScript variable definition corresponding to the specified ColdFusion variable value.

Category

Conversion functions, Extensibility functions

Function syntax

```
ToScript(cfvar, javascriptvar, outputformat, ASFormat)
```

See also

`cfwddx`; WDDX JavaScript Objects

History

ColdFusion MX 7: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfvar</td>
<td>A ColdFusion variable. This can contain one of the following:  &lt;br&gt;  • String  &lt;br&gt;  • Number  &lt;br&gt;  • Array  &lt;br&gt;  • Structure  &lt;br&gt;  • Query</td>
</tr>
<tr>
<td>javascriptvar</td>
<td>A string that specifies the name of the JavaScript variable that the ToScript function creates.</td>
</tr>
<tr>
<td>outputformat</td>
<td>Optional. A Boolean value that determines whether to create WDDX (JavaScript) or ActionScript style output for structures and queries:  &lt;br&gt;  • True: creates WDDX-style output (default).  &lt;br&gt;  • False: creates ActionScript-style output.</td>
</tr>
</tbody>
</table>
| **ASFormat** | Optional. A Boolean value that specifies whether to use ActionScript shortcuts in the script:  
|              | - False: does not use ActionScript shortcuts to create new Objects and new Arrays when generating the script. Instead, generates `New Object()` and `New Array()` in the script (default). |

**Usage**

To use a ColdFusion variable in JavaScript or ActionScript, the `ToScript` function must be in a `cfoutput` region and be surrounded by number signs (#). For example, the following code uses the `ToScript` function to convert a ColdFusion variable to a JavaScript variable:

```cfset```
```
thissString="hello world">
<script type="text/javascript" language="JavaScript">
     <cfoutput>
       var #toScript(thisString, "jsVar")#;
     </cfoutput>
</script>```

When ColdFusion runs this code, it sends the following to the client:

```script```
```
var jsVar = "hello world";
</script>```

An HTML script tag must enclose the JavaScript code. The `cfoutput` tag does not need to be inside the script block; it can also surround the block. WDDX-style output generates JavaScript code that creates a WDDXRecordset object, where the key of each record set entry is a column name, and the value of the recordList entry is an array of the corresponding query column entries, as follows:
WDDXQuery = new WddxRecordset();
col0 = new Array();
col0[0] = "John";
col0[1] = "John";
WDDXQuery["firstname"] = col0;
col0 = null;
coll = new Array();
coll[0] = "Lund";
coll[1] = "Allen";
WDDXQuery["lastname"] = coll;
coll = null;

To use WDDX-style output, first load the cf_webroot/CFIDE/scripts/wddx.js script, which defines JavaScript WDDX objects, as in the following line:

<script type="text/javascript" src="/CFIDE/scripts/wddx.js"> </script>

For more information on WDDX in JavaScript, see WDDX JavaScript Objects. ActionScript-style output generates code that creates an array of objects, where the array is indexed by row number, and the objects consist of column name - column value pairs, as follows:

ActionScriptQuery = new Array();
ActionScriptQuery[0] = new Object();
ActionScriptQuery[0]["firstname"] = "John";
ActionScriptQuery[0]["lastname"] = "Lund";
ActionScriptQuery[1] = new Object();
ActionScriptQuery[1]["firstname"] = "John";
ActionScriptQuery[1]["lastname"] = "Allen";

An ActionScript-style array does not require you to include the wddx.js file, and creates a variable that you can use in ActionScript on a Flash format form, for example, in an onChange attribute. If the outputformat parameter is False, setting ASFormat to True causes ToScript to use the ActionScript shortcut [] in place of New Array() and the shortcut {} in place of New Object(). Using these shortcuts allows you to pass ActionScript into cfform attributes without triggering ActionScript validation. If ASFormat is False, ToScript generates New Array() and New Object() in the script.

Example

The following example shows the results of converting a ColdFusion string, array, and query object to JavaScript variables. It also uses the string and array in JavaScript code.

<h2>ToScript</h2>
<h3>Converting a string variable</h3>
<cfset thisString = "This is a string">
<cfoutput>
<b>The thisString variable in ColdFusion</b><br>
#thisString#<br>
<br>
<strong>The output of ToScript(thisString, "jsVar")</strong><br>
#ToScript(thisString, "jsVar")#<br>
<br>
<cfoutput>
In a JavaScript script, convert thisString Variable to JavaScript <br>and output the resulting variable:</cfoutput><br>
<script type="text/javascript" language="JavaScript">
var #ToScript(thisString, "jsVar")#;
document.write("jsVar in JavaScript is: " + jsVar);
</script>
</cfoutput>

<h3>Converting an array</h3>
<!--- Create and populate a one-dimensional array --->
<cfset myArray=ArrayNew(1)>
<cfloop index="i" from="1" to="4">
  <cfset myArray[i]="This is array element" & i>
</cfloop>
<cfoutput>
<b>The ColdFusion myArray Array</b><br>
<!--- Write the contents of the myArray variable in ColdFusion --->
<cfloop index="i" from="1" to="#arrayLen(myArray)#">
  myArry[#i#]: #myArray[i]#<br>
</cfloop>
<br>
<strong>The output of ToScript(myArray, "jsArray")</strong><br>
#toScript(myArray, "jsArray")#<br>
<br>
<cfoutput>
In JavaScript, convert myArray to a JavaScript variable and write it's <br>contents</cfoutput><br>
<script type="text/javascript" language="JavaScript">
var #ToScript(myArray, "jsArray")#;
for (i in jsArray)
{
    document.write("myArray[" + i + "]: " + jsArray[i] + "+"<br>
);
}
</script>
<br>

<h3>Converting a query</h3>
This section converts the following query object to both WDDX format <br>and ActionScript type JavaScript objects.<br>

<cfquery name="thisQuery" datasource="cfdocexamples">
SELECT FirstName, LastName <br>FROM employee <br>WHERE FirstName = 'John'
</cfquery>

The Query in ColdFusion
<cftable query="thisQuery" headerlines="1" colheaders>
  <cfcol align="left" width="9" header="#FirstName#" text="#FirstName#"/>
<cftable align="left" width="9" header="<b>LastName</b>" text="#LastName#">
</cftable>

<cfcol align="left" width="9" header="<b>LastName</b>" text="#LastName#">
</cfcol>

<strong>JavaScript generated by ToScript(thisQuery, "WDDXQuery"):</strong><br>
#toScript(thisQuery, "WDDXQuery")#;<br>
<br>
<strong>JavaScript generated by ToScript(thisQuery, "ActionScriptQuery",
False):</strong><br>
#toScript(thisQuery, "ActionScriptQuery", False)#<br>
<br>
<!--- Convert to both WDDX format and ActionScript format --->
<script type="text/javascript" language="JavaScript">
#ToScript(thisQuery, "WDDXQuery")#;
#ToScript(thisQuery, "ActionScriptQuery", False)#;
</script>
<!--- For brevity, this example does not use JavaScript query variables --->
</cfoutput>
**ToString**

**Description**

Converts a value to a string.

**Returns**

A string.

**Category**

Conversion functions, String functions

**Function syntax**

```
ToString(value[, encoding])
```

**See also**

ToBase64, ToBinary, CharsetEncode; Using XML and WDDX in the Developing ColdFusion Applications

**History**

ColdFusion MX:

- Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode character values 0-65535. (ColdFusion 5 and earlier releases supported ASCII values 1-255.)
- Added the `encoding` parameter.
- Added ability to convert an XML document object to a string.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Value to convert to a string; can be a simple value such as an integer, a binary object, or an XML document object.</td>
</tr>
</tbody>
</table>
encoding

The character encoding (character set) of the string. Optional for binary data. Generates an error if used for a simple value or XML document object. The following list includes commonly used values:

- utf-8
- iso-8859-1
- windows-1252
- us-ascii
- shift_jis
- iso-2022-jp
- euc-jp
- euc-kr
- big5
- euc-cn
- utf-16

For more information on character encoding, see: [www.w3.org/International/O-charset.html](http://www.w3.org/International/O-charset.html). The default value is the encoding of the page on which the function is called. See `cfcontent`.

Usage

This function can convert simple values and binary values that do not contain Byte zero. If this function cannot convert a value, it throws an exception. This function can also convert an XML document object to a string XML representation. Adobe recommends that you use the `CharsetEncode` function to convert binary data to a string.

Example

```coldfusion
<h3>ToString Example</h3>
<!---- Initialize data. ---->
<cfset charData = ">">
<!----- Create string of ASCII characters (32-255) and concatenate them. ---->
<cfloop iIndex = "data" from = "32" to = "255">
  <cfset ch = chr(data)>
  <cfset charData = charData & ch>
</cfloop>
<p>The following string is the concatenation of characters (32 to 255) from the ASCII table.<br>
<cfoutput>#charData#</cfoutput></p>

<!----- Create a Base64 representation of this string. ---->
<cfset data64 = toBase64(#charData#)>
<p>The following string is the Base64 representation of the string.<br>
<cfoutput>#data64#</cfoutput></p>

<!----- Create a binary representation of Base64 data. ---->
<cfset dataBinary = toBinary(data64)>

<!----- Create the string representation of the binary data. ---->
<cfset dataString = ToString(dataBinary)>
<p>The following is the string representation of the binary data.<br>
<cfoutput>#dataString#</cfoutput></p>
```
Trace

Description

A function equivalent of the `<cftrace>` tag and is used in the `<cfscript>` mode.

Parameters

Same as the `<cftrace>` tag.

Category

Debugging functions

Function syntax

```
trace (var, text, type, category, inline, abort)
```

For positional notations, the sequence must be followed exactly in the same manner as provided in the syntax. If you do not provide one of the parameters, use an empty string instead. This does not apply to Boolean values for which you must provide proper values even if you have to skip them.

History

ColdFusion 11: Attributes passed to this function needs to be in a comma-separated format.

ColdFusion 9: Added this function.

See also

cfscript, cftrace

Usage

You can call this function as name=value pair or as positional argument.

Example
<cfscript>
function TotalInterest(principal, annualRate, months) {
    var years = 0;
    var interestRate = 0;
    var totalInterest = 0;
    principal = REReplace(trim(principal), "\$", ",", "ALL");
    annualRate = Replace(trim(annualRate), ",", ",", "ALL");

    if ((principal <= 0) OR (annualRate <= 0) OR (months <= 0)) {
        Throw(type="InvalidData",message="All values must be greater than 0.");
    } else {
        interestRate = annualRate / 100;
        years = months / 12;
        totalInterest = principal * (((1 + interestRate) ^ years) - 1);
        return DollarFormat(totalInterest);
    }
}

try {
    Trace(type="Information", inline="true", text="Calculating interest."); //Use of trace function in cfsript
    WriteOutput(TotalInterest("$2500.00", "5.5\%", "12");
    Trace(type="Information", inline="true", text="Interest calculation done.");
} catch(InvalidData ex) {
    //Displaying exception details on screen
    WriteOutput("<p>An InvalidData exception was thrown.</p>");
    WriteOutput("<p>#ex.message#</p>");
    //Writing the exception to log file under logs folder of web server.
    WriteLog(type="Error", file="myapp.log", text="#ex.type# #ex.message#";
}

From ColdFusion 11, attributes passed to this function needs to be in a comma-separated format.
**Transaction and concurrency**

When ORM methods are invoked without any transaction, all the data is committed to the database when the ORM session is flushed. ORM session is flushed when `ORMFlush()` is called or if autoflush is enabled when the request ends.

This works fine when there is not much concurrency, however, in most practical scenarios, you would need to use transaction in your application so that the data in your database is always in a consistent state.

With ColdFusion ORM, you can manage transactions in the following two ways:

- **Using Hibernate transaction**: User has full control, and ColdFusion does not intervene. The application has to flush/close the session and commit/rollback the transaction. For more information on transactions, go to the following URL: [http://community.jboss.org/wiki/sessionsandtransactions](http://community.jboss.org/wiki/sessionsandtransactions)

- **Using CFTransaction**: ColdFusion manages the transaction. Since a transaction cannot be distributed (across different data sources), the application must ensure that the changes made in the transaction affect only one Hibernate session. That is, only one data source. ColdFusion allows reading of data from other sessions (data source) in a transaction but changes must be made in only one session. Multiple dirty sessions at any time in the transaction can result in exceptions, and the transaction is rolled back. Before transaction begins, all existing sessions in the request are flushed.

  The previous session (if any) is reused. When the transaction is committed, the dirty session is automatically flushed (before committing the transaction). When the transaction is rolled back, the changed session cannot be used any longer because it can cause rolled back data to get committed later. Therefore, the session participating in the transaction is cleared when transaction is rolled back.

  A description of transaction is beyond the scope of this document. For more information on transactions, see the Hibernate documentation.

To run the ORM methods inside a transaction, they must be inside `<cftransaction>`. A simple example snippet of using ORM with `<cftransaction>` is as follows:

```coldfusion
<cftransaction>
  <cfset acct1 = EntityLoad("Account", "101")>
  <cfset acct2 = EntityLoad("Account", "102")>
  <cfset acct1.debit(1000)>
  <cfset acct2.credit(1000)>
  <cfset EntitySave(acct1)>
  <cfset EntitySave(acct2)>
</cftransaction>
```

Because we have not called commit on the `<cftransaction>` specifically, it is automatically committed when the `<cftransaction>` ends.

All `<cftransaction>` semantics including savepoint, multiple rollbacks, multiple commits, and nested transactions work with ORM. You can also have both queries and ORM in the same `<cftransaction>`.

When `<cftransaction>` begins, any existing ORM session is flushed and closed, and a new ORM session is created. The `<cftransaction>` can be committed or rolled back using appropriate ColdFusion tags in `<cftransaction>`. When the transaction ends and has not been committed or rolled back explicitly, it is automatically committed and the ORM session is closed. If there is any error inside the transaction, without any exception handling, the transaction is rolled back.

For more details on `<cftransaction>`, see the [CFML Reference Guide](https://adobe.github.io/cflib/).
Note

Even if ORMFlush() is called explicitly inside a <cftransaction> tag, the SQL runs but the data is committed only when the transaction commits.
TransactionCommit

Description

Commits the current active transaction.

Returns

Nothing

Category

Transaction functions

Function Syntax

TransactionCommit()

See also

TransactionRollback, TransactionSetSavePoint

History

ColdFusion 9: Added this function.

Usage

You can call this function only from within an active transaction.

Example
<cfscript>
q = new query();
q.setDatasource("cfartgallery");
WriteDump(q.execute(sql="select * from art where artid = 60").getResult());
transaction
{
    q.execute(sql="insert into art (artid,artistid,artname,description,price) values
        (60,3,'tom','tom',2000)");
    transactionSetSavePoint('sp1');
    WriteDump(q.execute(sql="select * from art where artid = 60").getResult());
    transaction
    {
        q.execute(sql="update art set artistid=4 where artid = 60");
        transactionSetSavePoint('sp2');
        WriteDump(q.execute(sql="select * from art where artid = 60").getResult());
        transaction
        {
            try
            {
                q.execute(sql="update art set artistid='badvalue' where artid = 60");
            }
            catch(any e)
            {
                WriteLog("rolling back the transaction");
                transactionRollback("sp1");
            }
        }
    }
}
WriteDump(q.execute(sql="select * from art where artid = 60").getResult());
transaction
{
    WriteLog("deleting the record");
    q.execute(sql="delete from art where artid = 60");
    WriteDump(q.execute(sql="select * from art where artid = 60").getResult());
}
</cfscript>
TransactionRollback

Description

Rolls back the current active transaction to the specified savepoint.

Returns

Nothing

Category

Transaction functions

Function Syntax

TransactionRollback(savepoint)

See also

TransactionCommit, TransactionSetSavePoint

History

ColdFusion 9: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>savepoint</td>
<td>An optional string identifier for the savepoint. Savepoints let you roll back portions of a transaction.</td>
</tr>
</tbody>
</table>

Usage

You can call this function only from within an active transaction. If you do not specify a savepoint, the current active transaction rolls back to the top-level (original) transaction.

Example

See the example for TransactionCommit.
TransactionSetSavePoint

Description

Creates and stores a new savepoint for the current transaction. You can add multiple savepoints by calling this function multiple times.

Returns

Nothing

Category

Transaction functions

Function Syntax

TransactionSetSavepoint (savepoint)<helponly></helponly>

See also

TransactionCommit, TransactionRollback

History

ColdFusion 9: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>savepoint</td>
<td>An optional string identifier for the savepoint. Savepoints let you roll back portions of a transaction.</td>
</tr>
</tbody>
</table>

Usage

You can call this function only from within an active transaction.

Example

See the example for TransactionCommit.
Trim

Description

Removes leading and trailing spaces and control characters from a string.

Returns

A copy of the string parameter, after removing leading and trailing spaces and control characters.

Category

String functions

Function syntax

Trim(string)

See also

LTrim, RTrim

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains a string.</td>
</tr>
</tbody>
</table>

Example

```
<h3>Trim Example</h3>
<cfif IsDefined("FORM.myText")>
  <cfoutput>
    <pre>
      Your string: "#FORM.myText#"
      Your string: "#Trim(FORM.myText)#"
      (trimmed on both sides)
    </pre>
  </cfoutput>
</cfif>
<form method = "post" action = "trim.cfm">
  <p>Type in some text, and it will be modified by trim to remove leading spaces from the left and right
  <p><input type = "Text" name = "myText" value = " TEST ">
  <p><input type = "Submit" name = "">
</form>
```
**UCase**

**Description**

Converts the alphabetic characters in a string to uppercase.

**Returns**

A copy of a string, converted to uppercase.

**Category**

String functions

**Function syntax**

```
UCase(string)
```

**See also**

LCase

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

**Example**

```
<h3>UCase Example</h3>

<cfif IsDefined("FORM.sampleText")>
  <cfif FORM.sampleText is not "">
    <p>Your text, <cfoutput>#FORM.sampleText#</cfoutput>, returned in uppercase is <cfoutput>#UCase(FORM.sampleText)#</cfoutput>.
  </cfif>
  <cfelse>
    <p>Please enter some text.</p>
  </cfif>
</cfif>

<form action = "ucase.cfm">
  <p>Enter your sample text, and press "submit" to see the text returned in uppercase:</p>
  <p><input type = "Text" name = "SampleText" value = "sample"></p>
  <input type = "Submit" name = "" value = "submit">
</form>
```
**URLDecode**

**Description**

Decodes a URL-encoded string.

**Returns**

A copy of a string, decoded.

**Category**

Conversion functions, Other functions, String functions

**Function syntax**

```
URLDecode(urlEncodedString[, charset])
```

**See also**

URLEncodedFormat; Tags and functions for globalizing applications in the Developing ColdFusion Applications

**History**

ColdFusion MX 6.1: Changed the default charset: the default charset is the character encoding of the URL scope.

ColdFusion MX:

- Changed Unicode support: ColdFusion supports the Java UCS-2 representation of Unicode character values 0-65535. (Earlier releases supported ASCII values.)
- Added the charset parameter.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>urlEncodedString</td>
<td>URL-encoded string or a variable that contains one.</td>
</tr>
<tr>
<td>charset</td>
<td>The character encoding in which the URL is encoded. Optional. The following list includes commonly used values:</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
|         | • utf-8  
|         | • iso-8859-1  
|         | • windows-1252  
|         | • us-ascii  
|         | • shift_jis  
|         | • iso-2022-jp  
|         | • euc-jp  
|         | • euc-kr  
|         | • big5  
|         | • euc-cn  
|         | • utf-16 |

For more information on character encoding, see: www.w3.org/International/O-charset.html. The default value is the character encoding of the URL scope.

**Usage**

URL encoding formats some characters with a percent sign and the two-character hexadecimal representation of the character. For example, a character whose code is 129 is encoded as %81. A space is encoded with a plus sign. Query strings in HTTP are always URL-encoded.

**Example**

This example creates, encodes, and decodes a string that contains ASCII character codes:

```cfscript
<cfscript>
// Build string
s = "";
for (c = 1; c le 256; c = c + 1)
{
    s = s & chr(c);
}
// Encode string and display result
enc = URLEncodedFormat(s);
WriteOutput("Encoded string is: '" & enc & "'.<br>");
// Decode and compare result with original
dec = URLDecode(enc);
if (dec neq s)
{
    WriteOutput("Decoded is not the same as encoded.");
}
else
{
    WriteOutput("All's quiet on the Western front.");
}
</cfscript>
```
URLEncodedFormat

Adobe recommends that you use the EncodeForURL function, not the URLEncodedFormat function, to escape special characters in a string for use in a URL in all new applications.

Description

Generates a URL-encoded string. For example, it replaces spaces with %20, and non-alphanumeric characters with equivalent hexadecimal escape sequences. Passes arbitrary strings within a URL (ColdFusion automatically decodes URL parameters that are passed to a page).

Returns

A copy of a string, URL-encoded.

Category

Conversion functions, Other functions, String functions

Function syntax

URLEncodedFormat(string [, charset ])

See also

URIDecode; Tags and functions for globalizing applications in the Developing ColdFusion Applications

History

ColdFusion MX 6.1: Changed the default encoding to be the response character encoding.

ColdFusion MX: Added the charset parameter.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>
**charset**

The character encoding in which the string is encoded.

Optional. The following list includes commonly used values:

- utf-8
- iso-8859-1
- windows-1252
- us-ascii
- shift_jis
- iso-2022-jp
- euc-jp
- euc-kr
- big5
- euc-cn
- utf-16

For more information on character encoding, see: [www.w3.org/International/O-charset.html](http://www.w3.org/International/O-charset.html). The default value is the character encoding of the response. See [cfcontent](#).

### Usage

URL encoding formats some characters with a percent sign and the two-character hexadecimal representation of the character. For example, a character whose code is 129 is encoded as `%81`. A space is replaced with `%20`. Query strings in HTTP are always URL-encoded.

### Example

```cfml
<h3>URLEncodedFormat Example</h3>
<cfif IsDefined("url.myExample")>
  <p>The url variable url.myExample was passed from the previous link ... its value is:
  &lt;br&gt;&lt;b&gt;&lt;cfoutput&gt;#url.myExample#&lt;/cfoutput&gt;&lt;/b&gt;</cfif>
<p>This function returns a URL encoded string.
<cfset s = "My url-encoded string has special characters & other stuff">
<p>&lt;A HREF = "urlencodedformat.cfm?myExample=#URLEncodedFormat(s)#"&gt;Click me&lt;/A&gt;</p>
```
**URLSessionFormat**

**Description**

Depending on whether a client computer accepts cookies, this function does the following:

- If the client does not accept cookies: automatically appends all required client identification information to a URL
- If the client accepts cookies: does not append information

This function automatically determines which identifiers are required, and sends only the required information. It provides a more secure and robust method for supporting client identification than manually encoding the information in each URL, because it sends only required information, when it is required, and it is easier to code.

**Returns**

A URL; if cookies are disabled for the browser, client and session data are appended.

**Category**

*Other functions*: Maintaining client identity in the *Developing ColdFusion Applications*

**Function syntax**

```
URLSessionFormat(request_URL)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request_URL</td>
<td>URL of a ColdFusion page</td>
</tr>
</tbody>
</table>

**Usage**

If the detected cookie is not turned on, or cookie support cannot be reliably detected. And if the *J2EE sessions* form of session variable management is enabled in the ColdFusion Administrator, the function return value is as follows:

```
myactionpage.cfm;jsessionid=xxxx (a semicolon is used instead of a question mark)
```

If the detected cookie is not turned on, or cookie support cannot be reliably detected, and if *ColdFusion session management* is used, the function return value is as follows:

```
myactionpage.cfm?CFID=xxxx&CFTOKEN=xxxxxxxxx
```
**J2EE session management** is used - Both *Enable Session Variables* and *Use J2EE session variables* are checked in the ColdFusion Administrator.

*ColdFusion session management* is used - Only *Enable Session Variables* is checked in ColdFusion Administrator (default).

### Example

```cfml
<cfform method="Post" action="#URLSessionFormat("MyActionPage.cfm")#">
</cfform>
```
Val

Description

Converts numeric characters that occur at the beginning of a string to a number.

Returns

A number. If conversion fails, returns zero.

Category

Conversion functions, String functions

Function syntax

Val(string)

See also

IsNumeric

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

Usage

This function works as follows:

- If TestValue = "234A56?7'", Val(TestValue) returns 234.
- If TestValue = "234'5678'9?'", Val(TestValue) returns 234.
- If TestValue = "BG234", Val(TestValue) returns the value 0, (not an error).
- If TestValue = "0", Val(TestValue) returns the value 0, (not an error).

Example
<h3>Val Example</h3>
<cfif IsDefined("FORM.theTestValue")>
  <cfif Val(FORM.theTestValue) is not 0>
    <h3>The string <cfoutput>#{DE(FORM.theTestValue)}#</cfoutput>
    can be converted to a number:
    <cfoutput>#{Val(FORM.theTestValue)}#</cfoutput></h3>
  </cfif>
<cfelse>
  <h3>The beginning of the string <cfoutput>#{DE(FORM.theTestValue)}#</cfoutput>
  cannot be converted to a number</h3>
</cfif>
</cfif>

<form action = "val.cfm">
<p>Enter a string, and determine whether its beginning can be evaluated to a numeric value.</p>
<p>
<input type = "Text"
       name = "TheTestValue"
       value = "123Boy">
<input type = "Submit"
       value = "Is the beginning numeric?"
       name = "">
</form>
ValueList

Description

Inserts a delimiter between each value in an executed query. ColdFusion does not evaluate the arguments.

Returns

A delimited list of the values of each record returned from an executed query.

Category

List functions, Query functions

Function syntax

```
ValueList(query.column [, delimiter ])
```

See also

QuotedValueList

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query.column</td>
<td>Name of an executed query and column. Separate query name and column name with a period.</td>
</tr>
<tr>
<td>delimiter</td>
<td>A delimiter character to separate column data items. The default value is comma (,).</td>
</tr>
</tbody>
</table>

Example
<h3>ValueList Example</h3>

<!--- use the contents of a query to create another dynamically --->
<cfquery name = "GetDepartments" datasource = "cfdocexamples">
  SELECT Dept_ID FROM Departments
  WHERE Dept_ID IN ('BIOL')
</cfquery>

<cfquery name = "GetCourseList" datasource = "cfdocexamples">
  SELECT *
  FROM CourseList
  WHERE Dept_ID IN ('#GetDepartments.Dept_ID#')
</cfquery>

Value list of all BIOL Course ID's using (--) as the delimiter:<br>
<cfoutput>
#ValueList(GetCourseList.Course_ID,"--")#<br>
</cfoutput>

Value list of all BIOL Course Numbers using (;) as the delimiter:<br>
<cfoutput>
#ValueList(GetCourseList.CorNumber,";")#<br>
</cfoutput>
VerifyClient

Description

Requires remote invocations of the page or calls to functions on the page to include an encrypted security token.

Returns

Does not return a value.

Category

Security functions

Function syntax

```
VerifyClient()
```

See also

cffunction, Improving security in Ajax programming rules and techniques in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function.

Parameters

Does not take any parameters

Usage

Use this function to help prevent security attacks where an unauthorized party attempts to perform an action on the server, such as changing a password. As a general rule, use this feature for Ajax requests to the server to perform sensitive actions, such as updating passwords. If you call this function, you must enable client management or session management in your application; otherwise, you do not get an error, but ColdFusion does not verify clients. Use this function only on pages that respond to client-side ColdFusion Ajax features, such as bind expressions. These features include code that correctly sends the security token when needed.
Week

Description

From a date/time object, determines the week number within the year.

Returns

An integer in the range 1-53; the ordinal of the week, within the year.

Category

Date and time functions

Function syntax

```
Week(date)
```

See also

DatePart

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date/time object in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

When passing date as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date.

Example

```html
<h3>Week Example</h3>
<cfif IsDefined("FORM.year")>
 More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
<cfoutput>
 <p>Your date, #DateFormat(yourDate)#. 
 <br>It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day 
 #DayOfWeek(yourDate)# in the week. 
 <br>This is day #Day(YourDate)# in the month of 
 #MonthAsString(Month(yourDate))#, which has #DaysInMonth(yourDate)# days. 
 <br>We are in week #Week(yourDate)# of #Year(yourDate)# (day 
 #DayOfYear(yourDate)# of #DaysInYear(yourDate)#). <br>
<cfif IsLeapYear(Year(yourDate))>This is a leap year 
<cfelse>This is not a leap year
</cfif>
</cfoutput>
</cfif>
</cfoutput>
</cfif>
```
Wrap

Description

Wraps text so that each line has a specified maximum number of characters.

⚠️ Note

The wrap function does not insert line breaks by placing the `<br>` tag in HTML text. Instead, it wraps the text in the display without adding the `<br>` tag.

Returns

String containing the wrapped text.

Category

String functions

Function syntax

```
Wrap(string, limit[, strip])
```

See also

cfmail

History

ColdFusion MX 6.1: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>String or variable that contains one. The text to wrap.</td>
</tr>
<tr>
<td>limit</td>
<td>Positive integer maximum number of characters to allow on a line.</td>
</tr>
<tr>
<td>strip</td>
<td>Boolean value specifying whether to remove all existing newline and carriage return characters in the input string with spaces before wrapping the text. The default value is False.</td>
</tr>
</tbody>
</table>

Usage

Inserts line break at the location of the first white space character (such as a space, tab, or new line) before the specified limit on a line. If a line has no whitespace characters before the limit, inserts a line break at the limit. Uses the operating-system specific line break: newline for UNIX, carriage return and newline on Windows. If you specify the `strip` parameter, all existing line breaks are removed, so any paragraph formatting is lost. Use this function to
limit the length of text lines, such as text to be included in a mail message. The `cfmail` and `cfmailpart` tag `wrap` text attributes use this function.

Example

```cfc
<h3>Wrap Example</h3>
<cfset inputText="This is an example of a text message that we want to wrap. It is rather long and needs to be broken into shorter lines.">
<cfoutput>#Wrap(inputText, 59)#</cfoutput>
```
**WriteDump**

**Description**

A function equivalent to the `<cfdump>` tag which can be used in `<cfscript>`.

**Parameters**

Same as `<cfdump>` tag.

**Category**

**Other functions, Data output functions**

**Function syntax**

```plaintext
WriteDump (var, output, format, abort, label, metainfo, top, show, hide, keys, expand, showUDFs);
```

**See also**

cfdump, cfscript

**History**

ColdFusion 11: Attributes passed to this function needs to be in a comma-separated format.

ColdFusion 9: Added this function.

**Usage**

You can call this function providing arguments as name=value pair or as positional arguments. For positional notations, the sequence must be followed exactly in the same manner as provided in the syntax. If you do not provide one of the parameters, use an empty string instead. This does not apply to Boolean values for which you must provide proper values even if you have to skip them.

**Example**

```cfs
<cfscript>
filename = "log.txt";
try {
    result = FileOpen(expandpath(filename));
    WriteDump(result);
} catch(Expression exception) {
    WriteOutput("<p>An Expression exception was thrown.</p>".ReadUInt32);
    WriteOutput("<p>#exception.message#</p>");
    WriteLog(type="Error", file="myapp.log", text="#exception.type# #exception.message#";
}
</cfscript>
```
From ColdFusion 11, attributes passed to this function needs to be in a comma-separated format.
WriteLog

Description

A function equivalent of the `cflog` tag, which can be used in `<cfscript>`.

Parameters

Same as the `cflog` tag:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Req/Opt</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Required</td>
<td></td>
<td>Message text to log.</td>
</tr>
</tbody>
</table>
| application | Optional | yes     | • yes: logs the application name, if it is specified in a `cfapplication` tag or `Application.cfc` file.  
• no |
| file      | Optional|         | Message file. Specify only the main part of the filename. For example, to log to the Testing.log file, specify "Testing". The file must be located in the default log directory. You cannot specify a directory path. If the file does not exist, it is created automatically, with the extension .log. |
| log       | Optional|         | If you omit the file attribute, writes messages to standard log file. Ignored, if you specify file attribute.  
• Application: writes to Application.log, normally used for application-specific messages.  
• Scheduler: writes to Scheduler.log, normally used to log the execution of scheduled tasks. |
<table>
<thead>
<tr>
<th>type</th>
<th>Optional</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Type (severity) of the message:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fatal</td>
</tr>
</tbody>
</table>

**Category**

**Data output functions**

**Function syntax**

```
WriteLog (text, type, application, file, log)
```

For positional notation, the sequence must be followed exactly in the same manner as provided in the syntax. If you do not provide one of the parameters, use an empty string instead. This does not apply to Boolean values for which you must provide proper values even if you have to skip them.

**See also**

cfscript, cflog

**Usage**

You can call this function as name=value pair or as positional argument.

**Example**
<cfscript>
//Example: 1
function TotalInterest(principal, annualRate, months) {
   var years = 0;
   var interestRate = 0;
   var totalInterest = 0;
   principal = REReplace(trim(principal), "[\$]", "", "ALL");
   annualRate = Replace(trim(annualRate), "%", "", "ALL");
   if ((principal <= 0) OR (annualRate <= 0) OR (months <= 0)) {
      Throw(type="InvalidData",message="All values must be greater than 0.");
   }
   interestRate = annualRate / 100;
   years = months / 12;
   totalInterest = principal * (((1 + interestRate) ^ years) - 1);
   return DollarFormat(totalInterest);
}

try {
   Trace(type="Information", inline="true", text="Calculating interest.");
   WriteOutput(TotalInterest("$2500.00", "5.5\%", "12");
   Trace(type="Information", inline="true", text="Interest calculation done.");
}

catch(InvalidData ex) {
   //Writting the exception to log file under logs folder of web server.
   WriteLog(type="Error", file="myapp.log", text="[#ex.type#] #ex.message#");
}
</cfscript>

From ColdFusion 11, attributes passed to this function needs to be in a comma-separated format.
WriteOutput

Description

Appends text to the page-output stream. This function writes to the page-output stream regardless of conditions established by the cfsetting tag.

Category

Other functions, System functions, Data output functions

Function syntax

WriteOutput(string)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
</tbody>
</table>

Usage

Within the cfquery and cfmail tags, this function does not output to the current page; it writes to the current SQL statement or mail text. Do not use WriteOutput within cfquery and cfmail. Although you can call this function anywhere within a page, it is most useful inside a cfscript block.

Example

```cfscript
...<cfscript>
employee = StructNew();
StructInsert(employee, "firstname", FORM.firstname);
StructInsert(employee, "lastname", FORM.lastname);
StructInsert(employee, "email", FORM.email);
StructInsert(employee, "phone", FORM.phone);
StructInsert(employee, "department", FORM.department);
WriteOutput("About to add " & FORM.firstname & " " & FORM.lastname);
</cfscript>```
**WSGetAllChannels**

**Description**

Sends messages to a specific channel based on the filter criteria (which is a struct).

**Returns**

Nothing

**History**

ColdFusion 10: Added this function

**Syntax**

```none
WSGetAllChannels()
WSGetAllChannels("channelName")
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>channelName</code></td>
<td>If specified, returns all sub-channels listed under the entered channel name. If left unspecified, the function returns all channels registered under the current application.</td>
</tr>
</tbody>
</table>
WSGetSubscribers

Description

Returns an array of struct with clientId and subscriberInfo as the keys.

Returns

Nothing

History

ColdFusion 10: Added this function

Syntax

```
WSGetSubscribers(channel)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel</td>
<td>Channel whose list of subscribers you wish to retrieve.</td>
</tr>
</tbody>
</table>
WSPublish

Description
Sends messages to a specific channel based on the filter criteria (which is a struct).

Category
Other functions, System functions, Data output functions

History
ColdFusion 10: Added this function

Syntax
WSPublish(String channel, Object message)
WSPublish(channel,message [,filterCriteria])

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel</td>
<td>Specific channel to which the server publishes its response.</td>
</tr>
<tr>
<td>message</td>
<td>Response sent by the server to all clients subscribed to a specific channel.</td>
</tr>
<tr>
<td>filterCriteria</td>
<td>Conditions to filter eligible clients that need to be notified for a given channel.</td>
</tr>
</tbody>
</table>

Example
<cfscript>
  if(isdefined("form.publish"))
  WsPublish(#form.channel#, #form.message#);
</cfscript>
<cfform method="post">
  <cfselect name="channel">
    <option>
      stocks
    </option>
    <option>
      news
    </option>
    <option>
      products
    </option>
  </cfselect>
  Message:
  <input id="message" name="message" type="text">
  <cfinput id="publish" name="publish" value="publish" type="submit">
</cfform>
WSSendMessage

Description

Sends messages to a specific client that invokes the method. This can be included as a part of the function that is called by the `invoke` WebSocket JavaScript method.

Returns

Nothing

History

ColdFusion 10: Added this function

Syntax

WSSendMessage(message)

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>Required. The message object. This can be of type Any.</td>
</tr>
</tbody>
</table>
**XmlChildPos**

**Description**

Gets the position of a child element within an XML document object.

**Returns**

The position, in an XmlChildren array, of the _N_ th child that has the specified name.

**Category**

XML functions

**Function syntax**

```plaintext
XmlChildPos(elem, childName, N)
```

**See also**

IsXmlElem, XmlElemNew, XmlSearch, XmlTransform; Using XML and WDDX in the Developing ColdFusion Applications

**History**

ColdFusion MX: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>elem</td>
<td>XML element within which to search.</td>
</tr>
<tr>
<td>childName</td>
<td>XML child element for which to search. Must be an immediate child of the elem parameter.</td>
</tr>
<tr>
<td>N</td>
<td>Index of XML child element for which to search.</td>
</tr>
</tbody>
</table>

**Usage**

You can use the returned index in the ArrayInsertAt and ArrayDeleteAt functions to change XML document objects. If the specified child is not found, the function returns -1.

**Example**

The following example searches XML document element, xmlObject.employee.name1, for its second Status element child and uses the position in an ArrayDeleteAt function to remove the element:
<!--- Create an XML document object --->
<cfxml variable="xmlobject">
<employee>
  <!-- A list of employees --->
  <name EmpType="Regular">
    <first>Almanzo</first>
    <last>Wilder</last>
    <Status>Medical Absence</Status>
    <Status>Extended Leave</Status>
  </name>
  <name EmpType="Contract">
    <first>Laura</first>
    <last>Ingalls</last>
  </name>
</employee>
</cfxml>

<!--- Find the second Status child of the first employee.name element --->
<cfscript>
  elempos=XMLChildPos(xmlobject.employee.name[1], "Status", 2);
  ArrayDeleteAt(xmlobject.employee.name[1].XmlChildren, elempos);
</cfscript>

<!--- Dump the resulting document object to confirm the deletion --->
<cfdump var="#xmlobject#"
**XmlElemNew**

**Description**

Creates an XML document object element.

**Returns**

An XML document object element.

**Category**

XML functions

**Function syntax**

```plaintext
XmlElemNew(xmlObj[, namespace], childName)
```

**See also**

cfxml, IsXmlElem, XmlChildPos, XmlFormat, XmlNew, XmlParse; Using XML and WDDX in the Developing ColdFusion Applications

**History**

ColdFusion MX 7: Added the namespace parameter.

ColdFusion MX: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlObj</td>
<td>Name of the XML document object in which you are creating the element</td>
</tr>
<tr>
<td>namespace</td>
<td>(Optional) URI of the namespace to which this element belongs</td>
</tr>
<tr>
<td>childName</td>
<td>Name of the element to create</td>
</tr>
</tbody>
</table>

**Usage**

The function's return variable specifies the location of the new element in the document object. It must specify a valid location in the document object identified by the `xmlObj` parameter. The following statements show this use:

```plaintext
ArrayAppend(MyDoc.MyRoot.XmlChildren, XmlElemNew(MyDoc,"childNode"));
```

If you do not specify a namespace URI and use a namespace prefix in the `childName` parameter, ColdFusion
checks to see if a namespace URI has already been specified for the prefix, and if so, uses that namespace.

Example

The following example creates and displays a ColdFusion document object:

```cfscript>
MyDoc = XmlNew();
MyDoc.xmlRoot = XmlElemNew(MyDoc,"MyRoot");
if (testVar IS TRUE)
    MyDoc.MyRoot.XmlText = "The value of testVar is True.";
else
    MyDoc.MyRoot.XmlText = "The value of testVar is False.";
for (i = 1; i LTE 4; i = i + 1)
    {
        MyDoc.MyRoot.XmlChildren[i].XmlText = "This is Child node " & i & ".";
    }
</cfscript>
<cfdump var=#MyDoc#>
Adobe recommends that you use the EncodeForXML function, not the XmlFormat function, to escape special characters in a string for use in XML in all new applications.

**Description**

Escapes special XML characters in a string so that the string can be used as text in XML.

**Returns**

A copy of the string parameter that is safe to use as text in XML.

**Category**

String functions, XML functions

**Function syntax**

```
XmlFormat(string, escapeChars)
```

**See also**

cfxml, XmlNew, XmlParse, XmlValidate; Using XML and WDDX in the Developing ColdFusion Applications

**History**

ColdFusion MX: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or a variable that contains one</td>
</tr>
<tr>
<td>escapeChars</td>
<td>Set to true to escape the characters restricted as per XML standards. For details, see <a href="http://www.w3.org/TR/2006/REC-xml11-20060816/#NT-RestrictedChar">http://www.w3.org/TR/2006/REC-xml11-20060816/#NT-RestrictedChar</a>.</td>
</tr>
</tbody>
</table>

**Usage**

This function escapes characters as follows:

<table>
<thead>
<tr>
<th>Text character</th>
<th>Escaped representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than symbol (&gt;)</td>
<td>&gt;</td>
</tr>
<tr>
<td>Less than symbol (&lt;)</td>
<td>&lt;</td>
</tr>
<tr>
<td>Single-quotiation mark (')</td>
<td>',</td>
</tr>
</tbody>
</table>
Double-quotation mark (")

Ampersand symbol (&)

Carriage return (but not line feed)

High ASCII characters in the range 159-255.

Example

The following example shows how XmlFormat escapes special XML characters. Use the View Source command in the browser to see the results. ColdFusion interprets the "" in the second text string as representing a single-quotation mark in text before it applies the XmlFormat function.

```
<?xml version = "1.0"?>
<cfoutput>
<someXML>
  <someElement someAttribute="#XmlFormat('a quoted value', "true")#">
    #XmlFormat("Body of element with <, >, ", " and & goes here.", "true")#
  </someElement>
</someXML>
</cfoutput>
```
**XmlGetNodeType**

**Description**

Determines the type of an XML document object node.

**Returns**

A string identifying the XML node type. The following values are valid:

<table>
<thead>
<tr>
<th>ATTRIBUTE_NODE</th>
<th>CDATA_SECTION_NODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENT_NODE</td>
<td>DOCUMENT_FRAGMENT_NODE</td>
</tr>
<tr>
<td>DOCUMENT_NODE</td>
<td>DOCUMENT_TYPE_NODE</td>
</tr>
<tr>
<td>ELEMENT_NODE</td>
<td>ENTITY_NODE</td>
</tr>
<tr>
<td>ENTITY_REFERENCE_NODE</td>
<td>NOTATION_NODE</td>
</tr>
<tr>
<td>PROCESSING_INSTRUCTION_NODE</td>
<td>TEXT_NODE</td>
</tr>
</tbody>
</table>

If the argument is not a document object node, the function generates an error.

**Category**

XML functions

**Function syntax**

```
XmlGetNodeType(xmlNode)
```

**See also**

IsXmlAttribute, IsXmlDoc, IsXmlElem, IsXmlNode, IsXmlRoot, XmlChildPos, XmlValidate; Using XML and WDDX in the Developing ColdFusion Applications

**History**

ColdFusion MX 7: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlNode</td>
<td>An XML DOM object node</td>
</tr>
</tbody>
</table>

**Usage**

The XmlGetNodeType function can determine the types of the nodes returned by the XmlSearch function, or the types of the entries in an element's XmlNodes array.
Example

The following example checks the node types of various parts of an XML document object:

<!---- Create an XML document object ---->
<cfxml variable="xmlobject">
    <?xml version="1.0" encoding="UTF-8"?>
    <order id="4323251">
        <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
        <items>
            <item id="43">
                <!-- This item is coded to show several node types -->
                <!--<![CDATA[Our Best hammer & chisel set!!!]]> Imported from France
                <quantity>1</quantity>
                <unitprice>15.95</unitprice>
            </item>
        </items>
    </order>
</cfxml>

<!---- Display the node types ---->
<cfoutput>
    <h3>Node Types</h3>
    xmlobject: #XMLGetNodeType(xmlobject)#<br>
    xmlobject.order: #XMLGetNodeType(xmlobject.order)#<br>
    Now check the types of all the nodes in the xmlobject.order.items.item element's XmlNodes array.<br>
    Note the many apparently empty Text nodes generated by whitespace characters in the XML text source.<br>
    <cfset descnodes=xmlobject.order.items.item.XmlNodes>
    <cfloop from="1" to="#ArrayLen(descnodes)#" index="i">
        #i# Node type is: #XMLGetNodeType(descnodes[i])#<br>
        #i# Node name is: #descnodes[i].XmlName#<br>
        <cfif (descnodes[i].XmlValue NEQ "")>
            #i# Node value is: #descnodes[i].XmlValue#<br>
        </cfif>
    </cfloop>
</cfoutput>
**XmlNew**

**Description**

Creates an XML document object.

**Returns**

An empty XML document object.

**Category**

XML functions

**Function syntax**

```
XmlNew([caseSensitive])
```

**See also**

cfxml, IsXmlDoc, ToString, XmlFormat, XmlParse, XmlValidate; Using XML and WDDX in the Developing ColdFusion Applications

**History**

ColdFusion MX: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>caseSensitive</td>
<td>Determines how ColdFusion processes the case of XML document object component identifiers:</td>
</tr>
<tr>
<td></td>
<td>• True: maintains case</td>
</tr>
<tr>
<td></td>
<td>• False: ColdFusion ignores case. Default.</td>
</tr>
</tbody>
</table>

**Usage**

An XML document object is represented in ColdFusion as a structure. The `caseSensitive` parameter value determines whether identifiers whose characters are of varying case, but are otherwise the same, refer to different components; for example:

- If True, the element or attribute names "name" and "NAME" refer to different elements or attributes.
- If False, these names refer to the same elements or attributes.

If your XML object is case sensitive, you cannot use dot notation to reference an element or attribute name. Use the name in associative array (bracket) notation, or a reference that does not use the case-sensitive name (such as xmlChildren1) instead. In the following code, the first line will work with a case-sensitive XML object. The second and third lines cause errors:
MyDoc.xmlRoot.XmlAttributes["Version"] = "12b";
MyDoc.xmlRoot.XmlAttributes.Version = "12b"
MyDoc.MyRoot.XmlAttributes["Version"] = "12b"

To convert an XML document object into a string, use the `ToString` function.

Example

The following example creates and displays a ColdFusion document object:

```cfc
<cfset testVar = True>
<cfscript>
MyDoc = XmlNew();
MyDoc.xmlRoot = XmlElemNew(MyDoc,"MyRoot");
if (testVar IS TRUE)
  MyDoc.MyRoot.XmlText = "The value of testVar is True."
else
  MyDoc.MyRoot.XmlText = "The value of testVar is False."
for (i = 1; i LTE 4; i = i + 1) {
  MyDoc.MyRoot.XmlChildren[i].XmlText = "This is Child node " & i & ".";
}
</cfscript>
<cfdump var=#MyDoc#>
XmlParse

Description

Converts XML text into an XML document object.

Returns

An XML document object.

Category

Conversion functions, XML functions

Function syntax

XmlParse(xmlText [, caseSensitive ], validator])

See also

cfxml, IsXML, ToString, XmlFormat, XmlNew, XmlSearch, XmlTransform, XmlValidate; Using XML and WDDX in the Developing ColdFusion Applications

History

ColdFusion MX 7:

- Added the validator parameter.
- Added support for filenames and URLs in the xmlText parameter.
- Added support for relative URLs and pathnames.
  ColdFusion MX: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlText</td>
<td>Any of the following:</td>
</tr>
<tr>
<td></td>
<td>• A string containing XML text.</td>
</tr>
<tr>
<td></td>
<td>• The name of an XML file.</td>
</tr>
<tr>
<td></td>
<td>• The URL of an XML file; valid protocol identifiers include http, https, ftp, and file.</td>
</tr>
<tr>
<td>caseSensitive</td>
<td>• Yes: maintains the case of document elements and attributes.</td>
</tr>
<tr>
<td></td>
<td>• No: Default</td>
</tr>
</tbody>
</table>
validator

<table>
<thead>
<tr>
<th>Any of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The name of a Document Type Definition (DTD) or XML Schema file.</td>
</tr>
<tr>
<td>• The URL of a DTD or Schema file; valid protocol identifiers include http, https, ftp, and file.</td>
</tr>
<tr>
<td>• A string representation of a DTD or Schema.</td>
</tr>
<tr>
<td>• An empty string; in this case, the XML file must contain an embedded DTD or Schema identifier, which is used to validate the document.</td>
</tr>
</tbody>
</table>

Usage

If you specify a relative URL or pathname in a parameter, ColdFusion uses the directory (or, for URLs, the logical directory) that contains the current ColdFusion page as the path root. The caseSensitive parameter value determines whether identifiers whose characters are of varying case, but are otherwise the same, refer to different components; for example:

- If true, the element or attribute names "name" and "NAME" refer to different elements or attributes.
- If false, these names refer to the same elements or attributes.

If your XML object is case sensitive, you cannot use dot notation to reference an element or attribute name. Use the name in associative array (bracket) notation, or a reference that does not use the case-sensitive name (such as xmlChildren1) instead. In the following code, the first line will work with a case-sensitive XML object. The second and third lines cause errors:

```plaintext
MyDoc.xmlRoot.XmlAttributes["Version"] = "12b";
MyDoc.xmlRoot.XmlAttributes.Version = "12b";
MyDoc.MyRoot.XmlAttributes["Version"] = "12b";
```

The optional validator parameter specifies a DTD or Schema to use to validate the document. If the parser encounters a validation error, ColdFusion generates an error and stops parsing the document. Specify a validator parameter to make the XMLParse function validate your document. If you do not specify a validator parameter, the XML file must specify the DTD or schema using the xsi:noNamespaceSchemaLocation tag. If you specify a validator parameter, also specify a caseSensitive parameter.

```plaintext
Note

If you specify an empty string as the third parameter to the XMLParse function and specify a validator such as xsi:noNamespaceSchemaLocation within an XML document, it must specify a complete URL. However, with XMLValidate, it can be just a filename that is found relative to the CFML template doing validation.
```

If you do not specify a validator parameter, the xmlText parameter can specify a well-formed XML fragment, and does not have to specify a complete document.

```plaintext
Note

To convert an XML document object back into a string, use the ToString function.
```

Example
The following example has three parts: an XML file, a DTD file, and a CFML page that parses the XML file and uses the DTD for validation. The CFML file displays the returned XML document object. To show the results of invalid XML, modify the bmenuD.xml.

⚠️ **Note**

The DTD used in the following example represents the same XML structure as the Schema used in the [XmlValidate](#) example.

The custorder.xml file is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE order SYSTEM "C:\ColdFusion\wwwroot\examples\custorder.dtd">
<order id="4323251">
  <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
  <items>
    <item id="43">
      <name> Deluxe Carpenter's Hammer</name>
      <quantity>1</quantity>
      <unitprice>15.95</unitprice>
    </item>
    <item id="54">
      <name> 36" Plastic Rake</name>
      <quantity>2</quantity>
      <unitprice>6.95</unitprice>
    </item>
    <item id="68">
      <name> Standard paint thinner</name>
      <quantity>3</quantity>
      <unitprice>8.95</unitprice>
    </item>
  </items>
</order>
```

The custorder.dtd file is as follows:

```xml
<!ELEMENT order (customer, items)>
<!ATTLIST order
  id CDATA #REQUIRED>
<!ELEMENT customer EMPTY>
<!ATTLIST customer
  firstname CDATA #REQUIRED
  lastname CDATA #REQUIRED
  accountNum CDATA #REQUIRED>
<!ELEMENT items (item+)>
<!ELEMENT item (name, quantity, unitprice)>
<!ATTLIST item
  id CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT quantity (#PCDATA)>
<!ELEMENT unitprice (#PCDATA)>
```
The CFML file is as follows. It uses a filename for the XML file and a URL for the DTD. Note that the XML and URL paths must be absolute.

```cfml
<cfset
myDoc=XMLParse("C:\ColdFusion\wwwroot\examples\custorder.xml",
false, "http://localhost:8500/examples/custorder.dtd")>
Dump of myDoc XML document object<br>
<cfdump var="#myDoc#">
```
**XmlSearch**

**Description**

Uses an XPath language expression to search an XML document object.

**Returns**

The results of the XPath search. For details, see Usage.

**Category**

XML functions

**Function syntax**

```
XmlSearch(xmlDoc, xPathString, params)
```

**See also**

cfxml, IsXML, XmlChildPos, XmlParse, XmlTransform; Using XML and WDDX in the Developing ColdFusion Applications

**History**

ColdFusion 10: Added support for XPath 2.0 syntax and for passing variables to XPath expressions.

ColdFusion 8: Added support for returning any valid XPath result, not just arrays of XML object nodes.

ColdFusion MX 7: Added support for attribute searches.

ColdFusion MX: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
params

Optional. A struct that contains key-value pair (variable name and value) document object as shown in the following example:

```cfscript
<cfscript>
    params = structNew();
    params["test"] = "food";
</cfscript>

<!--- find all nodes with element name as passed by variable test --->
<cfset result = xmlSearch(xmlDoc,"/breakfast_menu/*[local-name() eq $test]", params)>
```

xmlDoc
XML document object

XPathString
XPath expression

Usage

The `xmlSearch` function attempts to return the values returned by the search whenever possible. For example, if the XPath expression returns a Boolean, the CFML variable is assigned a `true` or `false` value. The following table lists XPath expression result data types and how they are represented in the CFML return value.

<table>
<thead>
<tr>
<th>XPath return type</th>
<th>ColdFusion representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Null</td>
<td>&quot;&quot; (empty string)</td>
</tr>
<tr>
<td>Number</td>
<td>Number</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>NodeSet</td>
<td>Array of XML nodes</td>
</tr>
<tr>
<td>Result Tree Fragment</td>
<td>Array of XML nodes</td>
</tr>
</tbody>
</table>

Results that are `Unknown` or have an unresolved variable in the expression throw an error. XPath is specified by the World Wide Web Consortium (W3C). For detailed information on XPath, including XPath expression syntax, see the W3C website at [www.w3.org/TR/xpath](http://www.w3.org/TR/xpath).

Example

The following example extracts the elements named `last`, which contain employee last names, from an XML file, and displays the names. The `employeesimple.xml` file contains the following XML:
The CFML file contains the following lines:

```cfscript
<cfscript>
    myxmlDoc = XmlParse("C:\CFusionMX7\wwwroot\examples\employeesimple.xml");
    selectedElements = XmlSearch(myxmlDoc, "/employee/name/last");
    for (i = 1; i LTE ArrayLen(selectedElements); i = i + 1)
        writeoutput(selectedElements[i].xmlText & "<br>");
</cfscript>`
**XmlTransform**

**Description**

Applies an Extensible Stylesheet Language Transformation (XSLT) to XML. The XML can be in string format or an XML document object.

**Returns**

A string containing the results of applying the XSLT to the XML.

**Category**

Conversion functions, XML functions

**Function syntax**

```coldfusion
XmlTransform(xml, xsl[, parameters])
```

**See also**

`cfxml`, `XmlFormat`, `XmlNew`, `XmlParse`, `XmlSearch`, `XmlValidate`; Using XML and WDDX in the Developing ColdFusion Applications

**History**

ColdFusion 10: Added support for XSLT 2.0 syntax

ColdFusion MX 7: Added the parameters parameter and the ability to use a file for the XSL.

ColdFusion MX: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml</td>
<td>An XML document in string format, or an XML document object</td>
</tr>
</tbody>
</table>
| xsl       | XSLT transformation to apply; can be any of the following: Any of the following:  
  - A string containing XSL text.  
  - The name of an XSTLT file. Relative paths start at the directory containing the current CFML page.  
  - The URL of an XSLT file; valid protocol identifiers include http, https, ftp, and file. Relative paths start at the directory containing the current CFML page. |
parameters

A structure containing XSL template parameter name-value pairs to use in transforming the document. The XSL transform defined in the xslString parameter uses these parameter values in processing the XML.

Usage

An XSLT converts an XML document to another format or representation by applying an Extensible Stylesheet Language (XSL) stylesheet to it. XSL, including XSLT syntax is specified by the World Wide Web Consortium (W3C). For detailed information on XSL and XSLT, see the W3C website at www.w3.org/Style/XSL. If the XSLT code contains include statements with relative paths, ColdFusion resolves them relative to the location of the XSLT file, or for an XSL string, the location of the current ColdFusion page.

Example

The following example converts an XML document that represents a customer order into an HTML document with the customer name and a table with the order items and quantities:

The custorder.xml file that represents a customer order has the following lines:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<order id="4323251">
  <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
  <items>
    <item id="43">
      <name>Deluxe Carpenter's Hammer</name>
      <quantity>1</quantity>
      <unitprice>15.95</unitprice>
    </item>
    <item id="54">
      <name>36&qout; Plastic Rake</name>
      <quantity>2</quantity>
      <unitprice>6.95</unitprice>
    </item>
    <item id="68">
      <name>Standard paint thinner</name>
      <quantity>3</quantity>
      <unitprice>8.95</unitprice>
    </item>
  </items>
</order>
```

The custorder.xsl XSLT file that transforms the XML to HTML that displays the customer's name, and the items and quantities ordered has the following lines:
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" doctype-public="-//W3C//DTD HTML 4.0 Transitional//EN"/>
  <xsl:template match="/">
    <html>
      <body>
        <table border="2" bgcolor="yellow">
          <tr>
            <th>Name</th>
            <th>Price</th>
          </tr>
          <xsl:for-each select="breakfast_menu/food">
            <tr>
              <td>
                <xsl:value-of select="name"/>
              </td>
              <td>
                <xsl:value-of select="price"/>
              </td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>

The CFML file has the following lines:

```cfml
<cffile action="read" file="C:\CFusionMX7\wwwroot\examples\custorder.xsl" variable="xmltrans">
<cfset xmldoc = XmlParse("C:\CFusionMX7\wwwroot\examples\custorder.xml")>
<cfoutput>(XmlTransform(xmldoc, xmltrans))</cfoutput>
```
**XmlValidate**

**Description**

Uses a Document Type Definition (DTD) or XML Schema to validate an XML text document or an XML document object.

**Returns**

The following validation structure:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>An array containing any validator error messages. These messages indicate that the document does not conform to the DTD or Schema (is not valid).</td>
</tr>
<tr>
<td>FatalErrors</td>
<td>An array containing any validator fatal error messages. Fatal errors indicate that the document contains XML formatting errors (is not well-formed XML).</td>
</tr>
<tr>
<td>Status</td>
<td>A Boolean value:</td>
</tr>
<tr>
<td>Warning</td>
<td>An array containing any validator warnings. A well-formed and valid document can produce warning messages.</td>
</tr>
</tbody>
</table>

**Category**

XML functions

**Function syntax**

```
XmlValidate(xmlDoc[, validator])
```

**See also**

cfxml, IsXmlDoc, IsXML, XmlFormat, XmlNew, XmlParse, XmlSearch, XmlTransform; Using XML and WDDX in the Developing ColdFusion Applications

**History**

ColdFusion MX 7: Added this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
xmlDoc

Any of the following:
- A string containing an XML document.
- The name of an XML file.
- The URL of an XML file; valid protocol identifiers include http, https, ftp, and file.
- An XML document object, such as one generated by the `XmlParse` function.

validator

Any of the following:
- A string containing a DTD or Schema.
- The name of a DTD or Schema file.
- The URL of a DTD or Schema file; valid protocol identifiers include http, https, ftp, and file.

Usage

If you specify a relative URL or filename in a parameter, ColdFusion uses the directory (or, for URLs, the virtual directory) that contains the current ColdFusion page as the path root. The `validator` parameter specifies a DTD or Schema to use to validate the document. If you omit the parameter, the XML document must contain one of the following:

- A `!DOCTYPE` tag to specify the DTD or its location
- An `xmlns:schemaLocation` or `xmlns:noNamespaceSchemaLocation` tag to specify the Schema location

If you use a `validator` parameter and the XML document specifies a DTD or Schema, the `XmlValidate` function uses the `validator` parameter, and ignores the specification in the XML document. If you do not use a `validator` parameter, and the XML document does not specify a DTD or Schema, the function returns a structure with an error message in the Errors field. This function attempts to process the complete XML document, and reports all errors found during the processing. As a result, the returned structure can have a combination of Warning, Error, and FatalError fields, and each field can contain multiple error messages.

Example

The following example has three parts: an XML file, an XSD Schema file, and a CFML page that parses the XML file and uses the Schema for validation. The CFML file displays the value of the returned structure's Status field and displays the returned structure. To show the results of invalid XML, modify the custorder.xml file.

⚠️ Note

The Schema used in the following example represents the same XML structure as the DTD used in the `XmlParse` example.

The custorder.xml file is as follows:
```xml
<?xml version="1.0" encoding="UTF-8"?>
<order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://localhost:8500/something.xsd" id="4323251" >
  <customer firstname="Philip" lastname="Cramer" accountNum="21"/>
  <items>
    <item id="43">
      <name> Deluxe Carpenter's Hammer</name>
      <quantity>1</quantity>
      <unitprice>15.95</unitprice>
    </item>
    <item id="54">
      <name> 36" Plastic Rake</name>
      <quantity>2</quantity>
      <unitprice>6.95</unitprice>
    </item>
    <item id="68">
      <name> Standard paint thinner</name>
      <quantity>3</quantity>
      <unitprice>8.95</unitprice>
    </item>
  </items>
</order>
```

The custorder.xsd file is as follows:
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
  <xs:element name="customer">
    <xs:complexType>
      <xs:attribute name="firstname" type="xs:string" use="required"/>
      <xs:attribute name="lastname" type="xs:string" use="required"/>
      <xs:attribute name="accountNum" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="quantity" type="xs:string"/>
  <xs:element name="unitprice" type="xs:string"/>
  <xs:element name="item">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="quantity"/>
        <xs:element ref="unitprice"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:integer" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="items">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="item" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="order">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="customer"/>
        <xs:element ref="items"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>

The CFML file is as follows. It uses a filename for the XML file and a URL for the Schema. The XML and URL paths must be absolute.
<cfset
myResults=XMLValidate("C:\CFusionMX7\wwwroot\examples\custorder.xml", "http://localhost:8500/examples/custorder.xsd")>
<cfoutput>
Did custorder.xml validate against custorder.xsd? #myResults.status#<br><br>
</cfoutput>
Dump of myResults structure returned by XMLValidate<br>
<cfdump var="#myResults#"/>
Year

Description

From a date/time object, gets the year value.

Returns

The year value of date.

Category

Date and time functions

Function syntax

```
Year(date)
```

See also

DatePart, IsLeapYear

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date/time object in the range 100 AD-9999 AD.</td>
</tr>
</tbody>
</table>

Usage

When passing a date as a string, enclose it in quotation marks. Otherwise, it is interpreted as a number representation of a date.

Example
<h3>Year Example</h3>

<cfif IsDefined("FORM.year")>
More information about your date:
<cfset yourDate = CreateDate(FORM.year, FORM.month, FORM.day)>
<cfoutput>
<p>Your date, #DateFormat(yourDate)#.
<br>It is #DayOfWeekAsString(DayOfWeek(yourDate))#, day #DayOfWeek(yourDate)# in the week.
<br>This is day #Day(yourDate)# in the month of #MonthAsString(Month(yourDate))#, which has #DaysInMonth(yourDate)# days.
<br>We are in week #Week(yourDate)# of #Year(YourDate)#.
<br>This is a leap year
</cfoutput>
<cfelse>
This is not a leap year
</cfif>
**YesNoFormat**

**Description**

Evaluates a number or Boolean value.

**Returns**

Yes, for a nonzero value; No for zero, `false`, and no Boolean values, and an empty string (""").

**Category**

*Decision functions, Display and formatting functions*

**Function syntax**

```
YesNoFormat(value)
```

**See also**

*IsBinary, IsNumeric*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A number or Boolean value</td>
</tr>
</tbody>
</table>

**Example**

```html
<h3>YesNoFormat Example</h3>
<p>The YesNoFormat function returns non-zero values as "Yes"; zero, false and no Boolean values, and empty strings (""") as "No".</p>

<cfoutput>
<ul>
<li>YesNoFormat(1): #YesNoFormat(1)#</li>
<li>YesNoFormat(0): #YesNoFormat(0)#</li>
<li>YesNoFormat("1123"): #YesNoFormat("1123")#</li>
<li>YesNoFormat("No"): #YesNoFormat("No")#</li>
<li>YesNoFormat(True): #YesNoFormat(True)#</li>
</ul>
</cfoutput>
```
Ajax JavaScript Functions

You can use the JavaScript functions listed below on pages that use ColdFusion Ajax features.

- Function summary Ajax
- ColdFusion.Ajax.submitForm
- ColdFusion.Autosuggest.getAutosuggestObject
- ColdFusion.Layout.enableSourceBind
- ColdFusion.MessageBox.getMessageBoxObject
- ColdFusion.MessageBox.isMessageBoxDefined
- ColdFusion.ProgressBar.getProgressBarObject
- JavaScriptFunctionsinColdFusion9Update1
## Function summary Ajax

The following table briefly describes the JavaScript functions that you can use in ColdFusion pages that use Ajax features:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColdFusion.Ajax.submitForm</td>
<td>Submits form data without refreshing the entire page when the results are returned.</td>
</tr>
<tr>
<td>ColdFusion.Autosuggest.getAutosuggestObject</td>
<td>Lets you access underlying YUI AutoComplete object thereby providing fine-grained control over the object, for example attaching an event.</td>
</tr>
<tr>
<td>ColdFusion.FileUpload.cancelUpload</td>
<td>Cancels the file upload at any point during the file upload.</td>
</tr>
<tr>
<td>ColdFusion.FileUpload.clearAllFiles</td>
<td>Clears all the files selected for upload.</td>
</tr>
<tr>
<td>Coldfusion.fileUpload.setUrl</td>
<td>Sets URL for the fileupload control dynamically.</td>
</tr>
<tr>
<td>ColdFusion.FileUpload.startUpload</td>
<td>Starts uploading the selected files.</td>
</tr>
<tr>
<td>ColdFusion.getInputElement</td>
<td>Gets the value of an attribute of a bindable ColdFusion control.</td>
</tr>
<tr>
<td>ColdFusion.grid.clearSelectedRows</td>
<td>Clears the selected rows in the grid.</td>
</tr>
<tr>
<td>ColdFusion.Grid.getBottomToolbar</td>
<td>Gets bottom toolbar that can be used to add a control, for example icon or button.</td>
</tr>
<tr>
<td>ColdFusion.Grid.getGridObject</td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified HTML $\text{cfgrid}$ control.</td>
</tr>
<tr>
<td>ColdFusion.grid.getSelectedRows</td>
<td>Fetches data for the selected rows in the grid.</td>
</tr>
<tr>
<td>ColdFusion.Grid.getTopToolbar</td>
<td>Gets the top toolbar that can be used to add a control, for example icon or button.</td>
</tr>
<tr>
<td>ColdFusion.Grid.hideBottomToolbar</td>
<td>Hides the bottom toolbar that can be used to add a control, for example icon or button.</td>
</tr>
<tr>
<td>ColdFusion.Grid.hideTopToolbar</td>
<td>Hides the top toolbar that can be used to add a control, for example icon or button.</td>
</tr>
<tr>
<td>ColdFusion.Grid.refresh</td>
<td>Manually refreshes a displayed grid.</td>
</tr>
<tr>
<td>ColdFusion.Grid.refreshBottomToolbar</td>
<td>Refreshes the bottom toolbar that can be used to add a control, for example icon or button.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>ColdFusion.Grid.refreshTopToolbar</code></td>
<td>Refreshes the top toolbar that can be used to add a control, for example icon or button.</td>
</tr>
<tr>
<td><code>ColdFusion.Grid.showBottomToolbar</code></td>
<td>Shows bottom toolbar that can be used to add a control, for example icon or button.</td>
</tr>
<tr>
<td><code>ColdFusion.Grid.showTopToolbar</code></td>
<td>Displays the top toolbar that can be used to add a control, for example icon or button.</td>
</tr>
<tr>
<td><code>ColdFusion.Grid.sort</code></td>
<td>Sorts the specified HTML grid.</td>
</tr>
<tr>
<td><code>ColdFusion.JSON.decode</code></td>
<td>Converts a JSON-encoded string into a JavaScript variable.</td>
</tr>
<tr>
<td><code>ColdFusion.JSON.encode</code></td>
<td>Converts a JavaScript variable into a JSON string.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.collapseAccordion</code></td>
<td>Collapses an area of an accordion layout.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.collapseArea</code></td>
<td>Collapses an area of a border layout (cflayout tag with a type attribute of border).</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.createAccordionPanel</code></td>
<td>Creates a panel in an existing accordion layout (cflayout tag with a type attribute of accordion).</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.createTab</code></td>
<td>Creates a tab in an existing tabbed layout (cflayout tag with a type attribute of tab).</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.disableSourceBind</code></td>
<td>Disables the source bind.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.disableTab</code></td>
<td>Disables the specified tab so it cannot be selected.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.enableSourceBind</code></td>
<td>If disabled, enables the source bind.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.enableTab</code></td>
<td>Enables the specified tab so users can select it and display the area contents.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.expandAccordion</code></td>
<td>Expands a collapsed area of an accordion layout.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.expandArea</code></td>
<td>Expands a collapsed area of a border layout.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.getAccordionLayout</code></td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified accordion type cflayout control.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.getBorderLayout</code></td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified border type cflayout control.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.getTabLayout</code></td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified tab type <code>cflayout</code> control.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.hideAccordion</code></td>
<td>Hides an accordion.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.hideArea</code></td>
<td>Hides a bordered layout area.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.hideTab</code></td>
<td>Hides a tab.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.selectAccordion</code></td>
<td>Selects an accordion and displays the layout area contents.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.selectTab</code></td>
<td>Selects a tab and displays the layout area contents.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.showAccordion</code></td>
<td>Shows an accordion that was hidden using the <code>initHide</code> attribute or the <code>hideArea()</code> function.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.showArea</code></td>
<td>Shows an area of a border layout that was hidden using the <code>initHide</code> attribute or the <code>hideArea()</code> function.</td>
</tr>
<tr>
<td><code>ColdFusion.Layout.showTab</code></td>
<td>Shows a tab that was hidden using the <code>initHide</code> attribute or the <code>hideTab()</code> function.</td>
</tr>
<tr>
<td><code>ColdFusion.Log.debug</code></td>
<td>Displays a debug-level message in the log window.</td>
</tr>
<tr>
<td><code>ColdFusion.Log.dump</code></td>
<td>Displays information about a complex variable in the log window.</td>
</tr>
<tr>
<td><code>ColdFusion.Log.error</code></td>
<td>Displays an error-level message in the log window.</td>
</tr>
<tr>
<td><code>ColdFusion.Log.info</code></td>
<td>Displays an information-level message in the log window.</td>
</tr>
<tr>
<td><code>ColdFusion.Map.addEvent</code></td>
<td>Enables event handling in a map.</td>
</tr>
<tr>
<td><code>ColdFusion.Map.addMarker</code></td>
<td>Adds a marker to the map.</td>
</tr>
<tr>
<td><code>ColdFusion.Map.getLatitudeLongitude</code></td>
<td>Gets the latitude/longitude coordinates for a given address.</td>
</tr>
<tr>
<td><code>ColdFusion.Map.getMapObject</code></td>
<td>Gets the Google map component.</td>
</tr>
<tr>
<td><code>ColdFusion.Map.hide</code></td>
<td>Hides the map if displayed.</td>
</tr>
<tr>
<td><code>ColdFusion.Map.refresh</code></td>
<td>Reloads the map.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>ColdFusion.Map.setCenter</code></td>
<td>Sets the center of map to the address that you specify.</td>
</tr>
<tr>
<td><code>ColdFusion.Map.setZoomlevel</code></td>
<td>Sets the zoom level of the map to the new value.</td>
</tr>
<tr>
<td><code>ColdFusion.Map.show</code></td>
<td>Shows the map if it is hidden.</td>
</tr>
<tr>
<td><code>ColdFusion.Mediaplayer.resize</code></td>
<td>Changes the current size of the media player.</td>
</tr>
<tr>
<td><code>ColdFusion.Mediaplayer.setMute</code></td>
<td>Mutes or unmutes the sound of the media player.</td>
</tr>
<tr>
<td><code>ColdFusion.Mediaplayer.setSource</code></td>
<td>Sets the URL of the FLV file.</td>
</tr>
<tr>
<td><code>ColdFusion.Mediaplayer.setVolume</code></td>
<td>Sets the volume of sound of the media player.</td>
</tr>
<tr>
<td><code>ColdFusion.Mediaplayer.startPlay</code></td>
<td>Plays the FLV file.</td>
</tr>
<tr>
<td><code>ColdFusion.Mediaplayer.stopPlay</code></td>
<td>Stops playing the FLV file.</td>
</tr>
<tr>
<td><code>ColdFusion.MessageBox.create</code></td>
<td>Creates a ColdFusion message box. Equivalent to the <code>&lt;cfmessagebox&gt;</code> tag.</td>
</tr>
<tr>
<td><code>ColdFusion.MessageBox.getMessageBoxObject</code></td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified HTML <code>&lt;cfmessagebox&gt;</code> control.</td>
</tr>
<tr>
<td><code>ColdFusion.MessageBox.isMessageBoxDefined</code></td>
<td>Checks if a message box is defined.</td>
</tr>
<tr>
<td><code>ColdFusion.MessageBox.show</code></td>
<td>Displays a ColdFusion message box.</td>
</tr>
<tr>
<td><code>ColdFusion.MessageBox.update</code></td>
<td>Updates message box properties.</td>
</tr>
<tr>
<td><code>ColdFusion.MessageBox.updateMessage</code></td>
<td>Updates the message property.</td>
</tr>
<tr>
<td><code>ColdFusion.MessageBox.updateTitle</code></td>
<td>Updates the message box title.</td>
</tr>
<tr>
<td><code>ColdFusion.navigate</code></td>
<td>Displays the output of a link URL in a specified <code>&lt;cfdiv&gt;</code>, <code>&lt;cflayoutarea&gt;</code>, <code>&lt;cfpod&gt;</code>, or <code>&lt;cfwindow&gt;</code> container.</td>
</tr>
<tr>
<td><code>ColdFusion.ProgressBar.getProgressBarObject</code></td>
<td>Gets the progress bar object.</td>
</tr>
<tr>
<td><code>ColdFusion.ProgressBar.hide</code></td>
<td>Hides the progress bar if it is displayed.</td>
</tr>
<tr>
<td><code>ColdFusion.ProgressBar.reset</code></td>
<td>Resets the progress.</td>
</tr>
<tr>
<td><code>ColdFusion.ProgressBar.show</code></td>
<td>Shows the progress bar if it is hidden.</td>
</tr>
<tr>
<td><code>ColdFusion.ProgressBar.start</code></td>
<td>Stops the underlying progress bar object that is running.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>ColdFusion.ProgressBar.stop</code></td>
<td>Starts the underlying progress bar object.</td>
</tr>
<tr>
<td><code>ColdFusion.ProgressBar.update</code></td>
<td>Updates the attributes <code>duration</code>, <code>interval</code>, and <code>oncomplete</code>.</td>
</tr>
<tr>
<td><code>ColdFusion.ProgressBar.updatestatus</code></td>
<td>Lets you manually update the status and message of the progress bar.</td>
</tr>
<tr>
<td><code>ColdFusion.setGlobalErrorHandler</code></td>
<td>Replaces the global JavaScript error handler for displaying information about ColdFusion Ajax errors.</td>
</tr>
<tr>
<td><code>ColdFusion.Slider.disable</code></td>
<td>Disables the slider control.</td>
</tr>
<tr>
<td><code>ColdFusion.Slider.enable</code></td>
<td>Enables the slider control.</td>
</tr>
<tr>
<td><code>ColdFusion.Slider.getSliderObject</code></td>
<td>Gets the slider control.</td>
</tr>
<tr>
<td><code>ColdFusion.Slider.getValue</code></td>
<td>Gets the numeric value of the slider control.</td>
</tr>
<tr>
<td><code>ColdFusion.Slider.hide</code></td>
<td>Hides the slider control.</td>
</tr>
<tr>
<td><code>ColdFusion.Slider.setValue</code></td>
<td>Sets the numeric value of the slider control.</td>
</tr>
<tr>
<td><code>ColdFusion.Slider.show</code></td>
<td>Shows the slider control.</td>
</tr>
<tr>
<td><code>ColdFusion.Tree.getTreeObject</code></td>
<td>Gets the underlying Yahoo YUI Library object for the specified HTML <code>cftree</code> control.</td>
</tr>
<tr>
<td><code>ColdFusion.Tree.refresh</code></td>
<td>Manually refreshes a displayed HTML tree.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.create</code></td>
<td>Creates a ColdFusion pop-up window. Equivalent to the <code>cfwindow</code> tag.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.getWindowObject</code></td>
<td>Gets the underlying Ext JS - JavaScript Library object for the specified HTML <code>cfwindow</code> control.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.hide</code></td>
<td>Hides a window</td>
</tr>
<tr>
<td><code>ColdFusion.Window.onHide</code></td>
<td>Specifies a JavaScript function to run each time a specific window hides.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.onShow</code></td>
<td>Specifies a JavaScript function to run each time a specific window shows.</td>
</tr>
<tr>
<td><code>ColdFusion.Window.show</code></td>
<td>Shows a hidden window.</td>
</tr>
</tbody>
</table>
ColdFusion.Ajax.submitForm

Description

Submits form data without refreshing the page when the results are returned.

Function syntax

ColdFusion.Ajax.submitForm(formId, URL[, callbackhandler, errorHandler, httpMethod, asynch])

See also

cfajaxproxy, ColdFusion.navigate, Using the ColdFusion.Ajax.submitForm function in Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formId</td>
<td>The ID or name attribute of the form.</td>
</tr>
<tr>
<td>URL</td>
<td>The URL to which to submit the form.</td>
</tr>
<tr>
<td>callbackhandler</td>
<td>The JavaScript function to handle a normal response. The function must take</td>
</tr>
<tr>
<td></td>
<td>a single argument, that contains the response body. This method is used</td>
</tr>
<tr>
<td></td>
<td>only if the form submission is asynchronous.</td>
</tr>
<tr>
<td>errorHandler</td>
<td>The JavaScript function to handle an HTTP error response. The function</td>
</tr>
<tr>
<td></td>
<td>must take two arguments: the HTTP status code, and the error message. This</td>
</tr>
<tr>
<td></td>
<td>method is used only if the form submission is asynchronous.</td>
</tr>
<tr>
<td>httpMethod</td>
<td>The HTTP method to use for the submission, must be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• GET</td>
</tr>
<tr>
<td></td>
<td>• POST (the default)</td>
</tr>
<tr>
<td>asynch</td>
<td>A Boolean value specifying whether to submit the form asynchronously. The</td>
</tr>
<tr>
<td></td>
<td>default value is true.</td>
</tr>
</tbody>
</table>
Returns

If the `asynch` argument is `false`, returns the response body. Otherwise, the function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function.

⚠️ Note

This function does not submit the contents of file fields.

Example

See `Using the ColdFusion.Ajax.submitForm function` in `Using Ajax form controls and features` in the `Developing ColdFusion Applications`. 

#back to top
ColdFusion.Autosuggest.getAutosuggestObject

Description

Lets you access underlying YUI AutoComplete object thereby providing fine-grained control over the object, for example attaching an event.

Returns

The underlying AutoComplete object.

Function syntax

ColdFusion.Autosuggest.getAutosuggestObject (Id)

Parameters

- Id: Name of the auto-suggest object.

Example
<html>
  <head>
    <cfajaximport tags="cfinput-autosuggest">
    <script>
      var init = function()
      {
        autosuggestobj = ColdFusion.Autosuggest.getAutosuggestObject('state');
        autosuggestobj.itemSelectEvent.subscribe(foo);
      }
      var foo = function(event, args)
      {
        var msg = "";
        msg = msg + "Event: " + event + "\n\n";
        msg = msg + "Selected Item: " + args[2] + "\n\n";
        msg = msg + "Index: " + args[1]._nItemIndex + "\n\n";
        alert(msg);
      }
      var getStates = function()
      {
        return ["California","Connecticut","Colorado","Illinois","Alabama","Iowa","Utah","Alaska"];
      }
    }
    </script>
  </head>
  <body>
    <h3>Attaching an event handler to the autosuggest object</h3>
    <cfform name="mycfform" method="post">
      State:<BR>
      <cfinput
        type="text"
        name="state"
        autosuggest="javascript:getStates({cfautosuggestvalue})"
        autosuggestMinLength=1
        autosuggestBindDelay=1>
      <cfset ajaxOnLoad("init")>
    </cfform>
  </body>
</html>
**ColdFusion.Chart.getChartHandle**

**Description**

Used to get chart object for various JavaScript operations that you want to perform, for example adding or removing a series or nodes in charts.

**Returns**

JavaScript object

**Function syntax**

```
ColdFusion.Chart.getChartHandle()
```

**History**

ColdFusion 10: Added this function

**Example**

The following example shows how to add a new value to the first series of a chart with chart ID `interactivebar`:

```
ColdFusion.Chart.getChartHandle().exec('interactivebar', 'appendseriesvalues', '{"plotindex": 0, "values" : [40]}' );
```

**Example**

The following example shows how to add a new value to the register a click event on a chart:

```
ColdFusion.Chart.getChartHandle().click = function(dataObj){
    alert("Chart Clicked - ID: " + data["id"]);
}
```

[#back to top](#)
ColdFusion.FileUpload.cancelUpload

Description
At any point during the upload of files, you can choose to cancel the upload. When you cancel the upload, any further upload of files stops.

Function syntax

ColdFusion.FileUpload.cancelUpload(name)

See also
ColdFusion.FileUpload.clearAllFiles, ColdFusion.FileUpload.startUpload

History
ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute in the cffileupload tag.</td>
</tr>
</tbody>
</table>

Returns
This function does not return a value.

Example
This example includes a button for the Cancel Upload action. When you click this button, the file upload is canceled.
<h3>This is an example of the FileUpload.cancelUpload function. Add a few files to upload and click Upload. During upload, click the Cancel Upload HTML button to cancel the upload.</h3>

```javascript
function onCancel()
    ColdFusion.FileUpload.cancelUpload('myupload');
};
</script>
<cffileupload
    url="uploadAll.cfm"
    progressbar="true"
    name="myupload"
    addButtonLabel = "Add File"
    clearButtonlabel = "Clear it"
    width=600
    height=400
    title = "File Upload"
    maxuploadsize="30"
    extensionfilter="*.jpg, *.png, *.flv, *.txt"
    BGCOLOR="#FFFFFF"
    MAXFILESELECT=10
    UPLOADBUTTONLABEL="Upload now"/>
<cfform name="form01">
    <cfinput type="button" name="cancelupld" value="Cancel Upload"
     onclick="onCancel()">
</cfform>
ColdFusion.FileUpload.clearAllFiles

Description

Before upload or after you cancel upload, you can choose to clear all the files selected for upload. When you do this, the selected files are removed from the Upload view list.

Function syntax

```coldfusion
ColdFusion.FileUpload.clearAllFiles(name)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the <code>name</code> attribute in the <code>cffileupload</code> tag.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Example

This example includes a button for the ClearAllFiles action. When you click this button, the files selected for upload are removed.
This is an example of the FileUpload.clearAllFiles function. Add a few files to upload and click Upload. During upload, click the Clear Upload HTML button to clear all the files selected for upload.

```javascript
function onClear() {
    ColdFusion.FileUpload.clearAllFiles('myupload');
}
</script>
<cffileupload url="uploadAll.cfm" progressbar="true" name="myupload" addButtonLabel = "Add File" clearButtonlabel = "Clear it" width=600 height=400 title = "File Upload" maxuploadsize="30" extensionfilter="*.jpg, *.png, *.flv, *.txt" BGCOLOR="##FFFFFF" MAXFILESELECT=10 UPLOADBUTTONLABEL="Upload now"/>
<cfinput type="button" name="clearupload" value="Cancel Upload" onclick="onClear()">
</cfform>
Coldfusion.fileUpload.setUrl

**Description**

Used to set URL for the fileupload control dynamically.

**Returns**

Nothing

**Function syntax**

ColdFusion.fileUpload.setUrl(id, url)

**Parameters**

- **Id**: Name of upload control.
- **Url**: URL can be an absolute URL, relative URL, or fully qualified URL.

**Example**
<script language="javascript">
var uploadDone = function(result){
    alert("File uploaded");
}

var setUploadUrl = function(id)
{
    var selectedFiles = ColdFusion.FileUpload.getSelectedFiles(id);
    var uploadUrl = "/manual/ajaxui/cffileupload/setUrl/includes/_uploadall.cfm";
    alert("Upload URL : " + uploadUrl);
    if(selectedFiles.length){
        ColdFusion.FileUpload.setURL(id,uploadUrl);
        ColdFusion.FileUpload.startUpload(id);
    }
}

var callbackhandler = function(obj)
{
    var fileName = obj["FILENAME"];
    var status = obj["STATUS"];
    var message = obj["MESSAGE"];
    var msg = "In callbackhandler()" + "\n\n" +
        "FILENAME: " + fileName + "\n\n" +
        "STATUS: " + status + "\n\n" +
        "MESSAGE: " + message
    alert(msg);
}

var errorhandler = function()
{
    alert("In errorhandler()");
}

var uploadcompleted = function()
{
    alert("All files have been uploaded successfully");
}
</script>
<cfform name="frmUpload">
<br>
<cffileupload name="uploader" hideuploadbutton="true" onComplete="uploadDone"
onError="errorhandler" onUploadComplete="uploadcompleted">
<br>
<cfinput type="button" name="submit" value="Click to set URL and Upload Files"
onClick="setUploadUrl('uploader')">
</cfform>
ColdFusion.FileUpload.startUpload

**Description**

Starts uploading the selected files.

**Function syntax**

```
ColdFusion.FileUpload.startUpload(name)
```

**See also**

[ColdFusion.FileUpload.cancelUpload](#), [ColdFusion.FileUpload.clearAllFiles](#)

**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the <code>name</code> attribute in the <code>cffileupload</code> tag.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return a value.

**Example**

This example includes a button to start the Upload action. When you click this button, the selected files start uploading.
<script language="JavaScript" type="text/javascript">
function onUpload() {
    ColdFusion.FileUpload.startUpload('myupload');
};
</script>
<cffileupload
url="uploadAll.cfm"
progressbar="true"
name="myupload"
addButtonLabel = "Add File"
clearButtonlabel = "Clear it"
width=600
height=400
title = "File Upload"
maxuploadsize="30"
extensionfilter="*.jpg, *.png, *.flv, *.txt"
BGCOLOR="##FFFFFF"
MAXFILESELECT=10
UPLOADBUTTONLABEL="Upload now"/>
<form name="form01">
<input type="button" name="startupload" value="Start Upload" onclick="onUpload()">
</form>
ColdFusion.getElementValue

Description

Gets the value of an attribute of a bindable ColdFusion control.

Function syntax

\[
\text{ColdFusion.getElementValue}(\text{elementId} [, \text{formId}, \text{attributeName}])
\]

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>elementId</td>
<td>The ID or name attribute of the control.</td>
</tr>
<tr>
<td>formId</td>
<td>The ID attribute of the form that contains the control. Omit this attribute if the element ID is unique on the page. If you omit this attribute and the element ID is not unique, the function uses the first element on the page with the specified ID.</td>
</tr>
<tr>
<td>attributeName</td>
<td>The control attribute to get; by default, the value attribute, or, for cfselect, the value of the selected element in the control. For cfgrid controls, use this attribute and specify the name of the column whose value you are getting; the function returns the entry in the currently selected row. For cftree controls, use this attribute and specify PATH or NODE. The function returns the item path or node value of the currently selected tree item.</td>
</tr>
</tbody>
</table>

Returns

The value of the specified attribute.

Usage

You can bind to, and get the attribute values of, the following HTML controls:

- cfgrid
- cfinput controls with checkbox, datefield, file, hidden, radio, or text types
- cfselect
- cftextarea
- cftree
ColdFusion.grid.clearSelectedRows

Description

Used to clear the selected rows in the grid.

Returns

Nothing

Function syntax

ColdFusion.grid.clearSelectedRows(id)

Parameters

- Id: Name of the grid defined using cfgrid.

Usage

See the following example.

Example

Employee.cfm

```html
<html>
<head>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<cfajaxproxy cfc="emp" jsclassname="emputils">
<cfscript language="javascript">
var emp = new emputils();
var deleteAllSelectedRows = function(grid)
{
    emp.setHTTPMethod("POST");
    emp.deleteEmployees(getAllSelectedRows(grid,false));
    ColdFusion.Grid.refresh(grid);
}
var getAllSelectedRows = function(grid,showalert)
{
    obj = ColdFusion.Grid.getSelectedRows(grid);
    jsonbj = ColdFusion.JSON.encode(obj);
    if(showalert)
        alert(jsonbj);
    return obj;
}
var clearAllSelectedRows = function(grid)
{
    ColdFusion.Grid.clearSelectedRows(grid);
}
</cfscript>
</head>
<body>
<cfform>
<cfgrid
format="html"
```
name="empListing"
selectmode="edit"

bind="cfc:emp.getEmployees({cfgridpage},{cfgridpagesize},{cfgridsortcolumn},{cfgridsortdirection})"
 onchange="cfc:emp.editEmployees({cfgridaction},{cfgridrow},{cfgridchanged})"
 autowidth="true"
 multirowselect=true
 delete="true"
 insert="true"
 title="Employee database"
 pagesize="25"
>
<cfgridcolumn name="EMP_ID" header="EMP_ID" select="false" display="false">
<cfgridcolumn name="FIRSTNAME" header="First Name" select="true" />
<cfgridcolumn name="LASTNAME" header="Last Name" select="true" />
<cfgridcolumn name="DEPARTMENT" header="Department" select="true" />
<cfgridcolumn name="EMAIL" header="Email" select="true" />
</cfgrid>
<br>
<cfinput type="button" onClick="javascript:getAllSelectedRows('empListing',true)"
 name="getRows" value="Get Selected Rows">
<cfinput type="button" onClick="javascript:clearAllSelectedRows('empListing')"
 name="clearRows" value="Clear Selected Rows">
<cfinput type="button" onClick="javascript:deleteAllSelectedRows('empListing')"
 name="deleteRows" value="Delete Selected Rows">
</cfform>
Employee.cfc

```coldfusion
cfcomponent
<cfscript>
    empQuery = new query(name="emps", datasource="cfdocexamples");
    remote any function
     getEmployees(page,pageSize,gridsortcolumn="EMP_ID",gridsortdirection="ASC",empName)
     {
         var orderBy = "EMP_ID";
         var mysql = "SELECT Emp_ID, FirstName, LastName, EMail, Department, Email FROM Employees";
         if(isdefined("arguments.empName") and trim(arguments.empName) neq ")
             mysql = mysql & " WHERE " & "firstname = '#arguments.empName#'";
         } if(arguments.gridsortcolumn eq ")
             mysql = mysql & " ORDER BY " & orderBy;
         } mysql = mysql & " ORDER BY " & gridsortdirection;
         return QueryConvertForGrid(empQuery.execute(sql=mysql).getResult(), page, pageSize);
     }
    remote void function editEmployees(gridaction,gridrow,gridchanged)
    {
        switch(gridaction)
        {
            case "I":
                { var eFName = gridrow['FIRSTNAME']; var eLName = gridrow['LASTNAME']; var eDept = gridrow['DEPARTMENT']; var eEmail = gridrow['EMAIL']; var insertSql = "insert into Employees(FirstName,LastName,Department,Email) values ('" & "' & eFName & "', '" & eLName & ", '" & eDept & "', '" & eEmail & ", '")";
                empQuery.execute(sql=insertSql);
                break;
            }
            case "U":
                { var empId = gridrow['EMP_ID']; var changedCol = structkeylist(gridchanged);
                var updateSql = "UPDATE Employees SET " & changedCol & ")" & gridchanged[changedCol] & " WHERE emp_id=" & empId;
                empQuery.execute(sql=updateSql);
                break;
            }
            case "D":
                { deleteEmployees(gridrow);
                }
        }
    }
    remote void function deleteEmployees(empdata)
```

{ 
  var i = 1;
  var emp = {};
  if(isArray(empdata) and not ArrayIsEmpty(empdata)){
    for(emp in empdata){
      if(isStruct(emp) and structkeyexists(emp,"emp_id")){
        empid = emp["emp_id"]; 
        writelog("deleting " & empid);
        //var deleteSql = "delete from Employees where emp_id=" & empid;
        //empQuery.execute(sql=deleteSql);
      }
    }
  }
}

In this example, setting `multirowselect=true` enables performing of batch operations on grid data, such as deleting multiple records. In the `deleteemployees` functions, two lines have been commented out to prevent accidental deletion of data (since it is a batch operation). To see deletion, uncomment the code. The form has a `deleteAllSelectedRows` button that illustrates how records can be deleted externally. That is, without using the delete button built in to the grid. The same approach can be used to perform other batch operations such as moving multiple files to another folder or batch updates.

⚠️ **Note**

Set the `httpMethod` to POST on the Proxy object carefully to avoid "request URI too large" errors as shown in the `deleteAllSelectedRows` method in `Employee.cfm`.

#back to top
ColdFusion.Grid.getBottomToolbar

Description

Gets bottom toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.getBottomToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar. ----

#back to top
ColdFusion.Grid.getGridObject

Description

Gets the underlying Ext (Ext JS JavaScript library) object for the specified HTML grid.

Function syntax

```plaintext
ColdFusion.Grid.getGridObject(name)
```

See also

cfgrid, ColdFusion_Grid.refresh, ColdFusion_Grid.sort, [Ext JS - JavaScript Library Documentation](http://extjs.com/deploy/ext/docs/], Using HTML grids in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute of the cfgrid tag for which you want the object.</td>
</tr>
</tbody>
</table>

Returns

If the grid is editable, an object of type Ext.gridEditableGrid; otherwise, an object of type Ext.grid.Grid.

Usage

Use this function to get the Ext toolkit (Ext.grid) object that underlies the ColdFusion HTML cfgrid control. You can then use the raw object to modify the displayed grid. For documentation on the objects and how to manage them, see the [Ext documentation](http://extjs.com/deploy/ext/docs/].
ColdFusion.grid.getSelectedRows

Description

Used to fetch data for the selected rows in the grid.

Returns

An array of objects that contains row data.

Function syntax

ColdFusion.grid.getSelectedRows(id)

Parameters

- Id: Name of the grid defined using cfgrid.

See also

FileUpload

Usage

See the example in ColdFusion.grid.clearSelectedRows.

Example

See the example in ColdFusion.grid.clearSelectedRows.

#back to top
ColdFusion.Grid.getTopToolbar

Description

Gets the top toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.getTopToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar.
ColdFusion.Grid.hideBottomToolbar

Description

Hides the bottom toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.hideBottomToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar.

#back to top
ColdFusion/Grid.hideTopToolbar

**Description**

Hides the top toolbar that can be used to add a control, for example icon or button.

**Function syntax**

ColdFusion/Grid.hideTopToolbar(Id)

**Parameters**

- Id: Name of the grid.

**Example**

See example in ColdFusion/Grid.refreshBottomToolbar.

#back to top
ColdFusion.Grid.refresh

Description

Manually refreshes a displayed grid.

Function syntax

ColdFusion.Grid.refresh(name [, preservePage])

See also

cfgrid, ColdFusion_Grid/GridObject, ColdFusion_Grid.sort, Ext
JS - JavaScript Library Documentation [http://extjs.com/deploy/ext/docs/], Using HTML grids in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute of the cfgrid tag to refresh.</td>
</tr>
<tr>
<td>preservePage</td>
<td>A Boolean value specifying whether to redisplay the current page of data (true), or display the first page of data (false, the default). This attribute applies only if the grid data requires multiple grid pages to display.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function is useful to refresh a grid when an event occurs that changes the underlying data but does not normally trigger a grid update.

Example

The following code snippet comes from an example that lets users delete rows from a grid. When the user selects a grid row and clicks the delete button, the Ajax proxy calls a mycfc.deleteRow function to delete the row from the database. When the function returns successfully, the proxy calls ColdFusion.Grid.refresh to update the grid and remove the row.
<cfajaxproxy bind="cfc:mycfc.deleteRow({deletebutton@click},{mygrid.id@none})" onSuccess="ColdFusion.Grid.refresh('mygrid', true)">

#back to top
ColdFusion.Grid.refreshBottomToolbar

Description

Refreshes the bottom toolbar that can be used to add a control, for example icon or button. This function internally calls the JavaScript function ColdFusion.Grid.showBottomToolbar.

Function syntax

ColdFusion.Grid.refreshBottomToolbar(Id)

Parameters

- Id: Name of the grid control.

Example

grid.cfc

```cfc
<cfcomponent>
<cfscript>
remotely any function
getEmployees(page, pageSize, gridsortcolumn="EMP_ID", gridsortdirection="ASC"){
    var startRow = (page-1)*pageSize;
    var endRow = page*pageSize;
    if(!isdefined("arguments.gridsortcolumn") or
        isdefined("arguments.gridsortcolumn") and trim(arguments.gridsortcolumn) eq "")
        gridsortcolumn = "EMP_ID";
    if(!isdefined("arguments.gridsortdirection") or
        isdefined("arguments.gridsortdirection") and arguments.gridsortdirection eq "")
        gridsortdirection = "ASC";
    var mysql = "SELECT Emp_ID, FirstName, EMail, Department FROM Employees";
    if(isdefined("arguments.gridsortcolumn") and
        arguments.gridsortcolumn neq "")
        mysql = mysql & " ORDER BY " & gridsortcolumn;
    if(isdefined("arguments.gridsortdirection") and
        arguments.gridsortdirection neq "")
        mysql = mysql & " " & gridsortdirection;
    rs1 = new query(name="team", datasource="cfdocexamples", sql=mysql).execute();
    return QueryConvertForGrid(rs1.getResult(), page, pageSize);
}

remote any function editEmployees(gridaction, gridrow, gridchanged){
    writelog("edit employee info");
}
</cfscript>
</cfcomponent>
```

grid.cfm
<script>
    var refreshToolbar = function(id,type){
        type == "top" ? ColdFusion.Grid.refreshTopToolbar(id) :
            ColdFusion.Grid.refreshBottomToolbar(id);
    }

    var hideToolbar = function(id,type){
        type == "top" ? ColdFusion.Grid.hideTopToolbar(id) :
            ColdFusion.Grid.hideBottomToolbar(id);
    }

    var showToolbar = function(id,type){
        (type == "top") ? ColdFusion.Grid.showTopToolbar(id) :
            ColdFusion.Grid.showBottomToolbar(id);
    }

    var handleToolbar = function(id,type){
        if(type == "top"){
            tbar = ColdFusion.Grid.getTopToolbar(id);
            tbar.addButton({
                text: "Add User Account",
                tooltip: "Add a user account",
                handler: addUserAccount
            });
        }
        else{
            bbar = ColdFusion.Grid.getBottomToolbar(id);
            bbar.add(new Ext.Toolbar.Separator());
            bbar.addButton({
                text: "Delete User Account",
                tooltip: "Delete a user account",
                handler: deleteUserAccount
            });
        }
    }

    var GetUserInfo = function(){
        alert("Retrieving user account");
    }

    var addUserAccount = function(){
        alert("Adding new user account")
    }

    var deleteUserAccount = function(){
        alert("Deleting user account")
    }
</script>
<cfform>
<br>
    <cfinput type="button" onClick="showToolbar('empGrid','top')" name="btn1"
        value="Show Top Toolbar">
    <cfinput type="button" onClick="handleToolbar('empGrid','top')" name="btn2"
        value="Add button to Top Toolbar">
    <cfinput type="button" onClick="refreshToolbar('empGrid','top')" name="btn3"
        value="Refresh Top Toolbar">
    <cfinput type="button" onClick="hideToolbar('empGrid','top')" name="btn4"
        value="Hide Top Toolbar">
</br><br>
<cfgrid
    format="html"
    name="empGrid"
    width="800"
    pagesize=5
    sort=true
    title="Employee database"
    collapsible="true"
    insert="yes"
    delete="yes"
    bind="cfc:grid.getEmployees({cfgridpage},{cfgridpagesize},{cfgridsortcolumn},{cfgridsortdirection})"
    onChange="cfc:grid.editEmployees({cfgridaction},{cfgridrow},{cfgridchanged})"
    selectMode="edit"
>
    <cfgridcolumn name="Emp_ID" display=false header="ID" />
    <cfgridcolumn name="FirstName" display=true header="First Name" />
    <cfgridcolumn name="Email" display=true header="Email" />
    <cfgridcolumn name="Department" display=true header="Department" />
</cfgrid>
<br><br>
<cfinput type="button" onClick="hideToolbar('empGrid','bottom')" name="btn5" value="Hide Bottom Toolbar">
<cfinput type="button" onClick="showToolbar('empGrid','bottom')" name="btn6" value="Show Bottom Toolbar">
<cfinput type="button" onClick="handleToolbar('empGrid','bottom')" name="btn7" value="Add button to Bottom Toolbar">
<cfinput type="button" onClick="refreshToolbar('empGrid','bottom')"
name="btn8" value="Refresh Bottom Toolbar">
</cfform>

#back to top
ColdFusion.Grid.refreshTopToolbar

Description

Refreshes the top toolbar that can be used to add a control, for example icon or button. This function internally calls the JavaScript function ColdFusion.Grid.showTopToolbar.

Function syntax

ColdFusion.Grid.refreshTopToolbar(Id)

Parameters

- **Id**: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar.

#back to top
ColdFusion.Grid.showBottomToolbar

Description

Shows bottom toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.showBottomToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar.

#back to top
ColdFusion.Grid.showTopToolbar

*Description*

Displays the top toolbar that can be used to add a control, for example icon or button.

*Function syntax*

```
ColdFusion.Grid.showTopToolbar(Id)
```

*Parameters*

- `Id`: Name of the grid.

*Example*

See example in `ColdFusion.Grid.refreshBottomToolbar`.

#back to top
ColdFusion.Grid.sort

Description
Sorts the specified HTML grid.

Function syntax

ColdFusion.Grid.sort(name [, columnName, direction])

See also
cfgrid, ColdFusion_Grid.getGridObject, ColdFusion_Grid.refresh. [Ext
JS - JavaScript Library Documentation](http://extjs.com/deploy/ext/docs/), Using HTML grids in the Developing
ColdFusion Applications

History
ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute of the cfgrid tag to sort.</td>
</tr>
<tr>
<td>columnName</td>
<td>The name of the column that determines the sort order.</td>
</tr>
<tr>
<td>direction</td>
<td>The sort direction. Must be one of these values:</td>
</tr>
<tr>
<td></td>
<td>• ASC (default)</td>
</tr>
<tr>
<td></td>
<td>• DESC</td>
</tr>
</tbody>
</table>

Returns
This function does not return a value.

Usage
This function sorts the data displayed by the grid by using a case-insensitive sort for string data, or a
numeric sort for numeric data. It uses the specified column contents to determine the displayed grid order.
When a grid has a remote data source, the bound CFC function that provides the data gets the column name and sort direction in the c
gridsortcolumn and cgridsortdirection bind attributes. The CFC function must use these values and perform the sort appropriately.

#back to top
ColdFusion.JSON.decode

Description

Converts a JSON-encoded string to a JavaScript variable.

Function syntax

ColdFusion.JSON.decode(string)

See also

ColdFusion.JSON.encode, DeserializeJSON, SerializeJSON, Using Ajax Data and Development Features in the Developing ColdFusion Applications, http://www.json.org

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string to encode.</td>
</tr>
</tbody>
</table>

Returns

A JavaScript variable containing the data in the JSON encoded string.

Usage

Use this function when you must explicitly convert between JavaScript and JSON format, for example, when you must call a remote function that is not in a CFC. If the JSON string has a security prefix as defined by the Server Settings > Settings page of the ColdFusion Administrator or specified in the cfapplication or cffunction tags, the function strips the prefix before decoding the string.

Example

The following example uses the ColdFusion.JSON.decode and ColdFusion.JSON.encode functions. When the user clicks the "Call" link, the callMe function encodes the String as JSON and calls the echo CFC's plainEcho function with the result. The function also sets the return format to plain, so that the CFC function does not automatically convert its return value to JSON, and sends plain text instead. The echo.cfc component has two functions:

- The plainEcho function converts its argument from JSON to a ColdFusion variable, calls the echo function, converts the result to JSON, and returns it to the caller.
- The echo function creates a structure, sets the structure's entry to the input parameter, and returns the result. (You could call this function remotely using to see the result of calling a function that does not encode JSON when you request a plain return type. To see the results, use the cfdebug HTTP parameter when you run the main page.) The main page has the following lines:
The echo.cfc file has the following lines:

```coldfusion
<cfcomponent output="false">

<cffunction name="echo" access="remote">
<cfargument name="text">
<cfset var ret = StructNew()>
<cfset ret.a = text>
<cfreturn ret>
</cffunction>

<cffunction name="plainEcho" access="remote">
<cfargument name="text">
<cfset t = deserializeJSON(text)>
<cfset ret = echo(t.A)>
<cfset ret = echo(t.A)>
<cfreturn serializeJSON(ret)>
</cffunction>

</cfcomponent>
```
ColdFusion.JSON.encode

Description

Converts a JavaScript expression to a JSON-encoded string.

Function syntax

```
ColdFusion.JSON.encode(expression)
```

See also

ColdFusion.JSON.decode, DeserializeJSON, SerializeJSON, Using Ajax Data and Development Features in the Developing ColdFusion Applications, http://www.json.org

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>An expression with the data to encode.</td>
</tr>
</tbody>
</table>

Returns

A String containing the data in JSON encoded format.

Usage

Use this function when you must explicitly convert between JavaScript and JSON format, for example, when you must call a remote function that is not in a CFC.

Example

See the example in ColdFusion.JSON.decode.

#back to top
ColdFusion.Layout.createTab

Description

Creates a tab and layout area in a ColdFusion tabbed layout.

Function syntax

```coldfusion
ColdFusion.Layout.createTab(layout, layoutArea, Title, URL [, configObject])
```

See also

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.disableTab`, `ColdFusion.Layout.enableTab`, `ColdFusion.Layout.showArea`, `ColdFusion.Layout.hideTab`, `ColdFusion.Layout.selectTab`, `ColdFusion.Layout.showTab`, `Using layouts in Controlling Ajax user interface layout` the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>The name attribute of the tabbed layout in which to add the tab.</td>
</tr>
<tr>
<td>layoutArea</td>
<td>The name to assign to the layout area that is created for the new tab. Must be unique on the page.</td>
</tr>
<tr>
<td>title</td>
<td>The text to display on the tab. You can use HTML mark-up to control the title appearance.</td>
</tr>
<tr>
<td>URL</td>
<td>The URL from which to get the layout area contents. This attribute can use URL parameters to pass data to the page. ColdFusion uses standard page path resolution rules to locate the page.</td>
</tr>
<tr>
<td>configObject</td>
<td>An object containing window configuration parameters. For details, see &quot;Usage&quot;.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function dynamically creates tabs in a tabbed layout; it is equivalent to putting a `cflayoutarea` tag inside a `cflayout` tag with a `type` attribute of `tab`. The `configuration` parameter defines tab characteristics; it can have any or all the following entries:
<table>
<thead>
<tr>
<th>Entry</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>align</strong></td>
<td>The <code>cflayout</code> tag <code>align</code> attribute value</td>
<td>Specifies how to align child controls within the layout area. The following values are valid:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• center</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• justify</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• left</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• right</td>
</tr>
<tr>
<td><strong>callbackhandler</strong></td>
<td></td>
<td>A function that is called when the layout tab body has loaded. This function must not take any arguments.</td>
</tr>
<tr>
<td><strong>closable</strong></td>
<td>false</td>
<td>A Boolean value specifying whether the user can close the window. If true, the tab has an X close icon.</td>
</tr>
<tr>
<td><strong>disabled</strong></td>
<td>false</td>
<td>A Boolean value specifying whether the tab is disabled, that is, whether user can select the tab to display its contents. Disabled tabs are grayed out. Ignored if there is a true selected entry.</td>
</tr>
<tr>
<td><strong>errorHandler</strong></td>
<td></td>
<td>A function that is called if an error occurs in loading the tab body. This function must take two arguments:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The HTTP status code, or -1 if the error is not an HTTP error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• An error message</td>
</tr>
<tr>
<td><strong>inithide</strong></td>
<td>false</td>
<td>A Boolean value specifying whether the tab is initially hidden. To show an initially hidden tab, use the ColdFusion.Layout.showTab function.</td>
</tr>
<tr>
<td>Property</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| overflow | auto  | Specifies how to display child content whose size would cause the tab layout area to overflow the window boundaries. The following values are valid:  
  - auto: Show scroll bars when necessary.  
  - hidden: Do not allow access to overflowing content.  
  - scroll: Always show horizontal and vertical scroll bars, even if they are not needed.  
  - visible: Content can display outside the bounds of the layout area.  
  
  **Note:** In Internet Explorer, layout areas with the visible setting expand to fit the size of the contents, rather than having the contents extend beyond the layout area. |
| selected | false | A Boolean value specifying whether this tab is initially selected so that its contents appears in the layout. |
| style    |       | A CSS style specification that controls the appearance of the layout area. |

**Example**

The following example creates a tabbed layout with one tab. When you click the button it creates a second tab that is immediately visible and selected. The main page looks as follows:
```html
<html>
<head></head>
<body>
<cfform name="layouts">
<cfinput type="button" name="CreateTab"
    onClick="ColdFusion.Layout.createTab('tabLayout', 'tab2',
        'Tab 2', '_tabUrl.cfm', {inithide: false, selected: true})"
    value="Create Tab">
</cfform>

<cflayout type="tab" name="tabLayout">
    <cflayoutarea name="tab1" title="Tab 1" align="left">
        Default Tab
    </cflayoutarea>
</cflayout>
</body>
</html>
```

The _tabUrl.cfm page looks as follows:

```html
<h3>Tab 2</h3>
This is a simple tab
```

#back to top
ColdFusion.Layout.collapseArea

Description
Collapses an area of a border layout.

Function syntax

```coldfusion
ColdFusion.Layout.collapseArea(layout, layoutArea)
```

See also
cflayout, cflayoutarea, ColdFusion.Layout.expandArea, ColdFusion.Layout.getTabLayout, ColdFusion.Layout.showArea, Using layouts in Controlling Ajax user interface layout the Developing ColdFusion Applications

History
ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>The name attribute of the border layout that contains the area to collapse.</td>
</tr>
<tr>
<td>layoutArea</td>
<td>The position in the layout of the area to collapse. Must be one of the following: bottom, left, right, or top.</td>
</tr>
</tbody>
</table>

Returns
This function does not return a value.

Usage
This function has no effect if the area is already collapsed.

Example
The following code snippet collapses the left area of the layout border layout when the user clicks the button.

```coldfusion
<cfinput name="collapse2" width="100" value="Collapse Area 2" type="button"
onClick="ColdFusion.Layout.collapseArea('thelayout', 'left');">
```

#back to top
ColdFusion.Layout.collapseAccordion

Description
Collapses an area of an accordion layout.

Function syntax
ColdFusion.Layout.collapseAccordion(layoutname, layoutareaname)

See also
the Developing ColdFusion Applications

History
ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layoutname</td>
<td>The name attribute of the accordion layout that contains the panel to collapse.</td>
</tr>
<tr>
<td>layoutareaname</td>
<td>The name of the panel in the accordion layout to collapse.</td>
</tr>
</tbody>
</table>

Returns
This function does not return a value.

Usage
This function has no effect if the accordion is already collapsed.

Example
The following code snippet collapses the accordion layout when the user clicks the button.

```<cfinput name="collapse2" width="100" value="Collapse Area 2" type="button" onClick="ColdFusion.Layout.collapseAccordion('thelayout', 'panel2')">```
ColdFusion.Layout.createAccordionPanel

Description

Creates a panel in a ColdFusion accordion layout.

Function syntax

```coldfusion
ColdFusion.Layout.createAccordionPanel(layoutname, layoutareaname, title, URL [, config])
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{{layout}}name</td>
<td>The name attribute of the accordion layout in which to add the panel.</td>
</tr>
<tr>
<td>{{layoutarea}}name</td>
<td>The name to assign for the new accordion panel. Must be unique on the accordion.</td>
</tr>
<tr>
<td>title</td>
<td>The text to display on the panel. You can use HTML mark-up to control the title appearance.</td>
</tr>
<tr>
<td>URL</td>
<td>The URL from which to get the panel area contents. This attribute can use URL parameters to pass data to the page. ColdFusion uses standard page path resolution rules to locate the page.</td>
</tr>
<tr>
<td>config</td>
<td>An object containing configuration parameters. For details, see &quot;Usage&quot;.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function dynamically creates panels in an accordion layout; it is equivalent to putting a cflayoutarea tag inside a cflayout tag with a type attribute of accordion. The configuration parameter defines panel
characteristics; it can have any or all the following entries:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>align</td>
<td>The <code>cflayout</code> tag <code>align</code> attribute value</td>
<td>Specifies how to align child controls within the panel area. The following values are valid:</td>
</tr>
</tbody>
</table>
|                |                              | • center  
|                |                              | • justify  
|                |                              | • left  
|                |                              | • right  
| callbackHandler|                              | A function that is called when the layout accordion body has loaded. This function must not take any arguments.                                                                                             |
| errorHandler   |                              | A function that is called if an error occurs in loading the tab body. This function must take two arguments:                                                                                              |
|                |                              | • The HTTP status code, or -1 if the error is not an HTTP error  
|                |                              | • An error message  
| overflow       | auto                         | Specifies how to display child content whose size would cause the tab layout area to overflow the window boundaries. The following values are valid:                                                    |
|                |                              | • auto: Show scroll bars when necessary.  
|                |                              | • hidden: Do not allow access to overflowing content.  
|                |                              | • scroll: Always show horizontal and vertical scroll bars, even if they are not needed.  
|                |                              | • visible: Content can display outside the bounds of the layout area.  
|                |                              | **Note:** In Internet Explorer, layout areas with the visible setting expand to fit the size of the contents, rather than having the contents extend beyond the layout area. |
| selected       | false                        | A Boolean value specifying whether this tab is initially selected so that its contents appears in the layout.                                                                                              |
| style | A CSS style specification that controls the appearance of the layout area. |
| titleicon | Specifies the location of the icon to display with the title. |

**Example**

The following example creates an accordion layout with one panel. When you click the button it creates a second panel that is immediately visible and selected. The main page looks as follows:

```html
<html>
<head>
</head>
<body>
<cfform name="panels">
  <cfinput type="button" name="CreateAccordionPanel"
    onClick="ColdFusion.Layout.createAccordionPanel('AccordionPanel','panel2','Panel 2','_panelUrl.cfm',{inithide:false,selected:true})"
    value="Create Panel">
</cfform>

<cflayout type="panel" name="AccordionPanel">
  <cflayoutarea name="panel1" title="Panel 1" align="left">
    Default Panel
  </cflayoutarea>
</cflayout>
</body>
</html>
```

The _tabURL.cfm page looks as follows:

```html
<h3>Panel 2</h3>
This is an accordion panel
```

*back to top*
ColdFusion.Layout.disableSourceBind

**Description**

Disables the source bind.

**Function syntax**

ColdFusion.Layout.disableSourceBind(Id)

**Parameters**

- **Id**: Name of the layout area.

**Usage**

Assume that you are using `Coldfusion.navigate` to populate content into tab or accordion panels. You can have instances where content comes from the source bind call if the `source` attribute is defined for `cflayoutarea` (and is not from `ColdFusion.navigate`). In such instances, you might disable the source bind to get content using `Coldfusion.navigate`.

**Example**

layout.cfm uses the templates Tab1_Src.cfm, Tab2_Src.cfm, and Tab3_Src.cfm. If you run layout.cfm, you notice that clicking

- `navigate` populates content of `tab2_src.cfm` instead of `navigate.cfm`
- Disable Source bind ensures that the content of `navigate.cfm` is populated in `tab2_src`
- Enable Source Bind and then clicking `tab2_src` would again populate the content of `tab2_src`

**Tab1_Src.cfm**

```cfdump var="#CGI#" keys="15" label="[CGI scope]"```

**Tab2_Src.cfm**

```cfdump var="#server#" label="[Server scope]"```

**Tab3_Src.cfm**

```cfdump var="#server.coldfusion#" label="[Showing key coldfusion in server scope]"```

**Tab4_Src.cfm**

```cfdump var="#server.os#" label="[Showing key OS in server scope]"```
layout.cfm

```html
<script>
var navigateToTab = function(layoutId, tabId){
    alert("Navigating to " + tabId);
    ColdFusion.Layout.selectTab(layoutId,tabId);
    ColdFusion.navigate('navigate.cfm',tabId);
}
var disableBind = function(tabId){
    alert("Disabling binding on source for " + tabId);
    ColdFusion.Layout.disableSourceBind(tabId);
}
var enableBind = function(tabId){
    alert("Enabling binding on source for " + tabId);
    ColdFusion.Layout.enableSourceBind(tabId);
}
</script>
<cflayout type="tab" name="layout1">
    <cflayoutarea
        name = "tab1"
        overflow = "auto"
        refreshonactivate = "yes"
        title = "Tab 1"
        source = "tab1_src.cfm"/>
    <cflayoutarea
        name = "tab2"
        overflow = "auto"
        refreshonactivate = "false"
        title = "Tab 2"
        source = "tab2_src.cfm"
        bindonload=false
    />
    <cflayoutarea
        name = "tab3"
        overflow = "auto"
        refreshonactivate = "yes"
        title = "Tab 3"
        source = "tab3_src.cfm"
    />
</cflayout>
<cfform name="myform">
    <cfinput type="button" name="disable" value="Disable Source Bind"
onClick="javascript:disableBind('tab2')">
    <cfinput type="button" name="b" value="Navigate"
onClick="javascript:navigateToTab('layout1','tab2')">
    <cfinput type="button" name="disable" value="Enable Source Bind"
onClick="javascript:enableBind('tab2')">
</cfform>
```

#back to top
ColdFusion.Layout.enableSourceBind

Description
If disabled, enables the source bind.

Function syntax
ColdFusion.Layout.enableSourceBind(1d)

Parameters
- 1d: Name of the layout area.

Usage
See usage in ColdFusion.Layout.disableSourceBind.

Example
See example in ColdFusion.Layout.disableSourceBind.

ColdFusion.FileUpload.getSelectedFiles

Description
Returns an array of objects containing the filename and size of the files selected for upload. The file size is returned in bytes. The function also returns file upload status as YES|NO|Error.

Function syntax
ColdFusion.FileUpload.getSelectedFiles(1d)

Parameters
- 1d: Name of the cffileupload control.

Usage
In a real life scenario, you normally use the uploader with other controls. For example, a form with three fields: name, email, and uploader. Assume that you upload the files, but forget to click Submit or you select the files, submit the form, but forget to click Upload. You can use this function to inform the user that there are files that have been selected for upload and provide the following details:

- FILENAME: Name of the file selected for upload.
- SIZE: Size of the file in bytes.
- STATUS: YES|NO|Error; YES indicates a successful upload, NO indicates that the upload is yet to occur, and Error indicates that an exception has occurred during the upload operation.

Example
The following example illustrates a scenario where the user clicks Submit and is informed about the files selected for upload:
<html>
<head>
<script language="javascript">
var formatNumber = function(num){
    if(num < 1024) return num + " bytes";
    if(num < (1024 * 1024)) return (num/1024).toFixed(2) + " KB";
    if(num < (1024 * 1024 * 1024)) return (num/(1024 * 1024)).toFixed(2) + " MB";
    return (num/(1024 * 1024 * 1024)).toFixed(2) + " GB";
}
var getSelectedList = function(id){
    var files = ColdFusion.FileUpload.getSelectedFiles(id);
    var fileslist = "";
    if(files.length)
        fileslist = "You have selected The following files for upload: \n\n";
    for(var i=0;i < files.length; i++){
        fileslist = fileslist + files[i].FILENAME + " (" + formatNumber(files[i].SIZE) + ")"
        if(i != files.length-1)
            fileslist = fileslist + "\r\n";
    }
    if(files.length)
        { alert(fileslist); }
}
</script>
</head>
<body>
<br>
<form name="frmUpload" method="POST">
    First Name: <cfinput type="text" name="fname" value=""> <br>
    Last Name: <cfinput type="text" name="lname" value=""> <br>
    <cfinput type="button" name="submitForm2" value="Submit"
        onClick="getSelectedList('myuploader1')">
</form>
</body>
</html>
ColdFusion.Layout.expandAccordion

Description
Expands a panel in an accordion layout.

Function syntax
ColdFusion.Layout.expandAccordion(layoutname, layoutareaname)

See also

History
ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layoutname</td>
<td>The name attribute of the accordion layout that contains the panel to expand.</td>
</tr>
<tr>
<td>layoutareaname</td>
<td>The name of the panel to expand. Must be one of the following: bottom, left, right, or top.</td>
</tr>
</tbody>
</table>

Returns
This function does not return a value.

Usage
This function has no effect if the panel is already expanded.

Example
The following code snippet expands the left area of the panel when the user clicks the button.

```<cfinput name="expand2" width="100" value="Expand Area 2" type="button"
onClick="ColdFusion.Layout.expandAccordion('thelayout', 'left');">```
ColdFusion.Layout.disableTab

Description

Disables the specified tab so it cannot be selected.

Function syntax

ColdFusion.Layout.disableTab(layout, layoutArea)

See also

cflayout, cflayoutarea, ColdFusion.Layout.createTab, ColdFusion.Layout.enableTab, ColdFusion.Layout.showArea, ColdFusion.Layout.hideTab, ColdFusion.Layout.selectTab, ColdFusion.Layout.showTab, Using layouts in Controlling Ajax user interface layout the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>The name attribute of the tabbed layout that contains the area to disable.</td>
</tr>
<tr>
<td>layoutArea</td>
<td>The name attribute of the tab layout area to disable.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function has no effect on the currently selected tab. A disabled tab appears grayed out.

Example

The following example lets you enable and disable a tab by clicking a link.
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>

<!--- The tabheight attribute sets the height of all tab content areas. --->
<cflayout type="tab" name="mainTab" tabheight="300px" style="width:400px">
  <cflayoutarea title="First Tab" name="tab1">
    <h2>The First Tab</h2>
    Here are the contents of the first tab.
  </cflayoutarea>
  
  <cflayoutarea title="Second Tab" name="tab2">
    <h2>The Second Tab</h2>
    This is the content of the second tab.
  </cflayoutarea>
</cflayout>

<p>
Use these links to test disabling/enabling via JavaScript. Note that you cannot disable the currently selected tab.<br />
<a href="" onClick="ColdFusion.Layout.enableTab('mainTab','tab1'); return false;">Click here to enable tab 1.</a><br />
<a href="" onClick="ColdFusion.Layout.disableTab('mainTab','tab1'); return false;">Click here to disable tab 1.</a><br />
</p>
</body>
</html>
### ColdFusion.Layout.enableTab

**Description**

Enables the specified tab so it can be selected.

**Function syntax**

```
ColdFusion.Layout.enableTab(layout, layoutArea)
```

**See also**

- `cflayout`, `cflayoutarea`, `ColdFusion.Layout.createTab`, `ColdFusion.Layout.disableTab`, `ColdFusion.Layout.showArea`, `ColdFusion.Layout.hideTab`, `ColdFusion.Layout.selectTab`, `ColdFusion.Layout.showTab`, `Using layouts in Controlling Ajax user interface layout`
- `Developing ColdFusion Applications`

**History**

ColdFusion 8: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>layout</code></td>
<td>The name attribute of the tabbed layout that contains the area to enable.</td>
</tr>
<tr>
<td><code>layoutArea</code></td>
<td>The name attribute of the tab layout area to enable.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return a value.

**Example**

See `ColdFusion.Layout.disableTab`

#back to top
ColdFusion.Layout.expandArea

Description
Expands an area of a border layout.

Function syntax

```
ColdFusion.Layout.expandArea(layout, layoutArea)
```

See also
cflayout, cflayoutarea, ColdFusion.Layout.collapseArea, ColdFusion.Layout.getTabLayout, ColdFusion.Layout.showArea, Using layouts in Controlling Ajax user interface layout the Developing ColdFusion Applications

History
ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>The name attribute of the border layout that contains the area to expand.</td>
</tr>
<tr>
<td>layoutArea</td>
<td>The position in the layout of the area to expand. Must be one of the following: bottom, left, right, or top.</td>
</tr>
</tbody>
</table>

Returns
This function does not return a value.

Usage
This function has no effect if the area is already expanded.

Example
The following code snippet expands the left area of the layout border layout when the user clicks the button.

```
<cfinput name="expand2" width="100" value="Expand Area 2" type="button"
onClick="ColdFusion.Layout.expandArea('thelayout', 'left');">  
```

#back to top
ColdFusion.Layout.getAccordionLayout

Description

Gets the underlying Ext (Ext JS JavaScript library) object for the specified accordion layout.

Function syntax

```
ColdFusion.Layout.getAccordionLayout(name)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute of the accordion type cflayout tag for which you want the object.</td>
</tr>
</tbody>
</table>

Returns

An object of type Ext.AccordionLayout.

Usage

Use this function to get the Ext toolkit (Ext.AccordionLayout) object that underlies the ColdFusion HTML cflayout control. You can then use the raw object to modify the displayed layout. For documentation on the objects and how to manage them, see the [Ext documentation](http://extjs.com/deploy/ext/docs/).
ColdFusion.Layout.getBorderLayout

Description

Gets the underlying Ext (Ext JS JavaScript library) object for the specified bordered layout.

Function syntax

ColdFusion.Layout.getBorderLayout(name)

See also

cflayout, cflayoutarea, ColdFusion.Layout.getTabLayout, [Ext JS
JavaScript Library Documentation][http://extjs.com/deploy/ext/docs/], Using layouts in Controlling Ajax user
interface layout the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute of the border type cflayout tag for which you want the object.</td>
</tr>
</tbody>
</table>

Returns

An object of type Ext.BorderLayout.

Usage

Use this function to get the Ext toolkit (Ext.BorderLayout) object that underlies the ColdFusion HTML cflayout control. You can then use the raw object to modify the displayed layout. For documentation on the objects and how to manage them, see the [Ext documentation][http://extjs.com/deploy/ext/docs/].
ColdFusion.Layout.getTabLayout

Description

Gets the underlying Ext (Ext JS JavaScript library) object for the specified tabbed layout.

Function syntax

```javascript
ColdFusion.Layout.getTabLayout(name)
```

See also

cflayout, cflayoutarea, ColdFusion.Layout.getBorderLayout, [Ext JS - JavaScript Library Documentation](http://extjs.com/deploy/ext/docs/), Using layouts in [Controlling Ajax user interface layout](http://extjs.com/deploy/ext/docs/) the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute of the border type cflayout tag for which you want the object.</td>
</tr>
</tbody>
</table>

Returns

An object of type Ext.BorderLayout.

Usage

Use this function to get the Ext toolkit (Ext.BorderLayout) object that underlies the ColdFusion HTML cflayout control. You can then use the raw object to modify the displayed layout. For documentation on the objects and how to manage them, see the [Ext documentation](http://extjs.com/deploy/ext/docs/).

#back to top
ColdFusion.Layout.hideAccordion

**Description**

Hides the specified panel and its accordion layout.

**Function syntax**

```
ColdFusion.Layout.hideAccordion(layoutname, layoutareaname)
```

**See also**


**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layoutname</td>
<td>The <code>name</code> attribute of the accordion layout that contains the panel to hide.</td>
</tr>
<tr>
<td>layoutareaname</td>
<td>The <code>name</code> attribute of the panel to hide.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return a value.

**Example**

The following example creates an accordion layout with two panels. Click the buttons to show and hide the second panel.
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>
<cflayout type="accordion" name="accordionLayout" accordionheight="300px" style="width:400px">
<cflayoutarea title="First Panel" name="panel1">
<h2>The First Panel</h2>
Here are the contents of the first panel.
</cflayoutarea>
<cflayoutarea title="Second Panel" name="panel2">
<h2>The Second Panel</h2>
This is the content of the second panel.
</cflayoutarea>
</cflayout>
<br />
<cfform name="layouts">
<cfinput type="button" name="ShowAccordion" value="Show Accordion"
onClick="ColdFusion.Layout.showAccordion('accordionLayout','panel2')">
<cfinput type="button" name="ShowAccordion" value="Hide Panel"
onclick="ColdFusion.Layout.hideAccordion('accordionLayout','panel2')">
</cfform>
</body>
</html>
ColdFusion.Layout.hideArea

Description

Hides an area of a border layout.

Function syntax

ColdFusion.Layout.hideArea(layout, layoutArea)

See also

cflayout, cflayoutarea, ColdFusion.Layout.collapseArea, ColdFusion.Layout.expandArea, ColdFusion.Layout.showArea, Using layouts in Controlling Ajax user interface layout in Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>The name attribute of the border layout that contains the area to hide.</td>
</tr>
<tr>
<td>layoutArea</td>
<td>The position in the layout of the area to hide. Must be one of the following: bottom, left, right, or top.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function has no effect if the area is already hidden.

Example

The following code snippet hides the left area of the layout border layout when the user clicks the button.

```
<cfinput name="hide2" width="100" value="Hide Area 2" type="button"
onClick="ColdFusion.Layout.hideArea('thelayout', 'left');">
```

#back to top
**ColdFusion.Layout.hideTab**

*Description*

Hides the specified tab and its layout area.

*Function syntax*

```csharp
ColdFusion.Layout.hideTab(layout, layoutArea)
```

*See also*

`cflayout`, `cflayoutarea`, `ColdFusion.Layout.createTab`, `ColdFusion.Layout.disableTab`, `ColdFusion.Layout.enableTab`, `ColdFusion.Layout.selectTab`, `ColdFusion.Layout.showTab`, `Using layouts` in **Controlling Ajax user interface layout** the *Developing ColdFusion Applications*

*History*

ColdFusion 8: Added this function

*Parameters*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>The name attribute of the tabbed layout that contains the area to hide.</td>
</tr>
<tr>
<td>layoutArea</td>
<td>The name attribute of the tab layout area to hide.</td>
</tr>
</tbody>
</table>

*Returns*

This function does not return a value.

*Example*

The following example creates a layout with two tabs. Click the buttons to show and hide the second tab.
Here are the contents of the first tab.

This is the content of the second tab.

#back to top
**ColdFusion.Layout.selectAccordion**

**Description**

Selects the specified accordion layout and displays its panels.

**Function syntax**

```
ColdFusion.Layout.selectAccordion(layoutname, layoutareaname)
```

**See also**


**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layoutname</td>
<td>The name attribute of the accordion layout that contains the area to select.</td>
</tr>
<tr>
<td>layoutareaname</td>
<td>The name attribute of the panel to select.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return a value.

**Usage**

This function has no effect on a disabled panel.

**Example**

The following code lets you select each of the two panels in an accordion layout.
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>
    <cflayout type="accordion" name="mainAccordion" accordionheight="300px" style="width:400px">
        <cflayoutarea title="First Panel" name="panel1">
            <h2>The First Panel</h2>
            Here are the contents of the first panel.
        </cflayoutarea>

        <cflayoutarea title="Second Panel" name="panel2">
            <h2>The Second Panel</h2>
            This is the content of the second panel.
        </cflayoutarea>
    </cflayout>

    <p>
    Use these links to test selecting tabs via JavaScript:<br />
    <a href="" onClick="ColdFusion.Layout.selectAccordion('mainAccordion','panel1'); return false;">Click here to select panel 1.</a><br />
    <a href="" onClick="ColdFusion.Layout.selectAccordion('mainAccordion','panel2'); return false;">Click here to select panel 2.</a><br />
    </p>
</body>
</html>
ColdFusion.Layout.selectTab

Description

Selects the specified tab and displays its layout area.

Function syntax

```
ColdFusion.Layout.selectTab(layout, layoutArea)
```

See also

cflayout, cflayoutarea, ColdFusion.Layout.createTab, ColdFusion.Layout.disableTab, ColdFusion.Layout.enableTab, ColdFusion.Layout.hideTab, ColdFusion.Layout.showTab, Using layouts in Controlling Ajax user interface layout the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>The name attribute of the tabbed layout that contains the area to select.</td>
</tr>
<tr>
<td>layoutArea</td>
<td>The name attribute of the tab layout area to select.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function has no effect on a disabled tab.

Example

The following code lets you select each of the two tabs in a layout.
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>
<cflayout type="tab" name="mainTab" tabheight="300px" style="width:400px">
<cflayoutarea title="First Tab" name="tab1">
<h2>The First Tab</h2>
Here are the contents of the first tab.
</cflayoutarea>
<cflayoutarea title="Second Tab" name="tab2">
<h2>The Second Tab</h2>
This is the content of the second tab.
</cflayoutarea>
</cflayout>
<p>Use these links to test selecting tabs via JavaScript:<br />
<a href="" onClick="ColdFusion.Layout.selectTab('mainTab','tab1'); return false;">Click here to select tab 1.</a><br />
<a href="" onClick="ColdFusion.Layout.selectTab('mainTab','tab2'); return false;">Click here to select tab 2.</a><br />
</p>
</body>
</html>
ColdFusion.Layout.showAccordion

Description

Shows a panel in the accordion layout that was hidden using the `coldfusion.layout.hideAccordion` function.

Function syntax

```
ColdFusion.Layout.showAccordion(layoutname, layoutareaname)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layoutname</td>
<td>The <code>name</code> attribute of the accordion layout that contains the panels to show.</td>
</tr>
<tr>
<td>layoutareaname</td>
<td>The <code>name</code> attribute of the accordion layout area whose panels you want to show.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function shows only the panel of an accordion layout area; it does not show the display area. To show the display area of a hidden panel, call this function, followed by ColdFusion.Layout.selectAccordion. This function does not show a panel that a user closed by clicking the x icon on the panel.

Example

See ColdFusion.Layout.hideAccordion.

#back to top
ColdFusion.Layout.showArea

Description

Shows an area of a border layout that was hidden by using the cflayoutarea tag inithide attribute or the ColdFusion.Layout.hideArea() JavaScript function.

Function syntax

```plaintext
ColdFusion.Layout.showArea(layout, layoutArea)
```

See also

cflayout, cflayoutarea, ColdFusion.Layout.collapseArea, ColdFusion.Layout.expandArea, ColdFusion.Layout.getTabLayout, Using layouts in Controlling Ajax user interface layout the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>The name attribute of the border layout that contains the area to show.</td>
</tr>
<tr>
<td>layoutArea</td>
<td>The position in the layout of the area to show. Must be one of the following: bottom, left, right, or top.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function does not show an area that a user closed by clicking the x icon on the title bar. Other areas move, if needed, to accommodate the area. This function has no effect if the area is already visible.

Example

The following code snippet shows the left area of the layout border layout when the user clicks the button.

```html
<cfinput name="show2" width="100" value="Show Area 2" type="button"
onClick="ColdFusion.Layout.showArea('thelayout', 'left');">
```

#back to top
ColdFusion.Layout.showTab

Description

Shows a tab that was hidden by using the initHide attribute of the cflayoutarea tag or the hideTab() JavaScript function.

Function syntax

ColdFusion.Layout.showTab(layout, layoutArea)

See also

cflayout, cflayoutarea, ColdFusion.Layout.createTab, ColdFusion.Layout.disableTab, ColdFusion.Layout.enableTab, ColdFusion.Layout.hideTab, ColdFusion.Layout.selectTab, Using layouts in Controlling Ajax user interface layout the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layout</td>
<td>The name attribute of the tabbed layout that contains the tab to show.</td>
</tr>
<tr>
<td>layoutArea</td>
<td>The name attribute of the tab layout area whose tab you want to show.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function shows only the tab of a layout area; it does not show the display area. To show the display area of a hidden tab, call this function, followed by ColdFusion.Layout.selectTab. This function does not show a tab that a user closed by clicking the x icon on the tab.

Example

See ColdFusion.Layout.hideTab.

#back to top
**ColdFusion.Log.debug**

*Description*

Displays a debug-level message in a log window.

*Function syntax*

```
ColdFusion.Log.debug(message [, category])
```

*See also*

`ColdFusion.Log.dump`, `ColdFusion.Log.error`, `ColdFusion.Log.info`. Logging information in Debugging Ajax applications the Developing ColdFusion Applications

*History*

ColdFusion 8: Added this function

*Parameters*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The text message to display in the log window. The log message can include HTML markup and JavaScript variables.</td>
</tr>
<tr>
<td>category</td>
<td>A category identifier that you can use in the logging window to filter the output. You can specify any arbitrary category in this function. The default value is <code>global</code>.</td>
</tr>
</tbody>
</table>

*Returns*

This function does not return a value.

*Usage*

If the page that calls this function does not have any ColdFusion AJAX-based controls, you use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function. The log window appears if you specify a URL parameter of the format `cfdebug` or `cfdebug=true` in your page request and you select the Enable Ajax Debug Log Window option on the ColdFusion Administrator Debugging & Logging > Debug Output Settings page.

*Example*

```
```

#back to top
ColdFusion.Log.dump

Description
Displays a debug-level message in the log window that shows a cfdump-like representation of a complex JavaScript object. The log window does not have a separate dump level.

Function syntax

ColdFusion.Log.dump(object [, category])

See also
ColdFusion.Log.debug, ColdFusion.Log.error, ColdFusion.Log.info, Logging information in Debugging Ajax applications the Developing ColdFusion Applications

History
ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The variable whose contents you want to display. You cannot specify additional contents, such as a text message, when you dump a complex object. To provide additional information, also use the ColdFusion.Log.debug function.</td>
</tr>
<tr>
<td>category</td>
<td>A category identifier that you can use in the logging window to filter the output. You can specify any arbitrary category in this function. The default value is global.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, you use a cfajaximport tag on the page to ensure that the page includes the JavaScript definition for this function. The log window appears if you specify a URL parameter of the format cfdebug or cfdebug="true" in your page request and you select the Enable Ajax Debug Log Window option on the ColdFusion Administrator Debugging & Logging > Debug Output Settings page.

Example
ColdFusion.Log.dump(objArg, "Pod A");
ColdFusion.Log.error

Description
Displays an error-level message in a log window.

Function syntax

ColdFusion.Log.error(message [, category])

See also
ColdFusion.Log.debug, ColdFusion.Log.dump, ColdFusion.Log.info, Logging information in Debugging Ajax applications the Developing ColdFusion Applications

History
ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The text message to display in the log window. The log message can include HTML markup and JavaScript variables.</td>
</tr>
<tr>
<td>category</td>
<td>A category identifier that you can use in the logging window to filter the output. You can specify any arbitrary category in this function. The default value is global.</td>
</tr>
</tbody>
</table>

Returns
This function does not return a value.

Usage
If the page that calls this function does not have any ColdFusion AJAX-based controls, you use a cfajaximport tag on the page to ensure that the page includes the JavaScript definition for this function. The log window appears if you specify a URL parameter of the format cfdebug or cfdebug="true" in your page request and you select the Enable Ajax Debug Log Window option on the ColdFusion Administrator Debugging & Logging > Debug Output Settings page.

Example

ColdFusion.Log.error("<b>Invalid value:</b><br>" + arg.A, "Pod A");
ColdFusion.Log.info

Description
Displays an information-level message in a log window.

Function syntax

ColdFusion.Log.info(message [, category])

See also

ColdFusion.Log.debug, ColdFusion.Log.dump, ColdFusion.Log.error, Logging information in Debugging Ajax applications the Developing ColdFusion Applications

History
ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The text message to display in the log window. The log message can include HTML markup and JavaScript variables.</td>
</tr>
<tr>
<td>category</td>
<td>A category identifier that you can use in the logging window to filter the output. You can specify any arbitrary category in this function. The default value is global.</td>
</tr>
</tbody>
</table>

Returns
This function does not return a value.

Usage
If the page that calls this function does not have any ColdFusion AJAX-based controls, you use a cfajaximport tag on the page to ensure that the page includes the JavaScript definition for this function. The log window appears if you specify a URL parameter of the format cfdebug or cfdebug="true" in your page request and you select the Enable Ajax Debug Log Window option on the ColdFusion Administrator Debugging & Logging > Debug Output Settings page.

Example

ColdFusion.Map.addEvent

Description

Executes a custom JavaScript function to enable event handling in a map.

Function syntax

```javascript
ColdFusion.Map.addEvent(name, event, listener, scopeObject)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the map.</td>
</tr>
<tr>
<td>event</td>
<td>The event to handle, for example click, dblclick, singleRightClick and mapTypeChange. For more events, refer to the Events section in the Google Maps API Reference documentation.</td>
</tr>
<tr>
<td>listener</td>
<td>The function that is called when the event is fired.</td>
</tr>
<tr>
<td>scopeObject</td>
<td>A JavaScript object that is set in the this scope.</td>
</tr>
</tbody>
</table>

Usage

This function does not return a value.

Example
<h3>This is an example of the Map.addMarker function. Click the HTML button labeled "Add marker" to set the marker to the specified Address.</h3>
<script>
var markerObj={
    address: '201 S. Division St. Suite 500 Ann Arbor, MI 48104'
};
function addmarker(){
    ColdFusion.Map.addMarker('mapID', markerObj);
}
</script>
<h3>MAP 1</h3>
<cfform name="map01">
<cfinput type="button" value="Add marker" name="markerbutton" onclick="javascript:addmarker();">
</cfform>
<cfmap name="mapID"
centerlatitude=42.261
centerlongitude=-87.717
displayscale=true
doubleclickzoom="true"
overview=true
scrollwheelzoom=true
tips="My Map" zoomlevel="4">
</cfmap>
ColdFusion.Map.addMarker

Description

Adds a marker to the map.

Function syntax

ColdFusion.Map.addMarker(name, markerObj)

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the <code>name</code> attribute of the <code>cfmap</code> tag.</td>
</tr>
<tr>
<td>markerObj</td>
<td>Specifies the marker object for a given address. The following are the associated properties: <code>latitude</code>, <code>longitude</code>, <code>address</code>, <code>title</code>, <code>markerColor</code>, <code>markerIcon</code>, <code>address</code>, <code>markerWindowContent</code>, and <code>showMarkerWindow</code>.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Example
<h3>This is an example of the Map.addmarker function. Click the HTML button labeled "Add marker" to set the marker to the specified Address.</h3>

```html
var markerObj={
    address: '201 S. Division St. Suite 500 Ann Arbor, MI 48104'
};
function addmarker(){
    ColdFusion.Map.addMarker('mapID', markerObj);
}
</script>
<h3>MAP 1</h3>
<cfform name="map01">
    <cfinput type="button" value="Add marker" name="markerbutton" onclick="javascript:addmarker();">
</cfform>
<cfmap name="mapID"
    centerlatitude=42.261
    centerlongitude=-87.717
    displayscale=true
    doubleclickzoom="true"
    overview=true
    scrollwheelzoom=true
    tips="My Map" zoomlevel="4">
</cfmap>
ColdFusion.Map.getLatitudeLongitude

Description

Gets the latitude/longitude coordinates for a given address.

Function syntax

ColdFusion.Map.getLatitudeLongitude("address", "callBack")

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>Specifies the value in the address attribute of the cfm ap tag.</td>
</tr>
<tr>
<td>callBack</td>
<td>The callback function that runs after the longitude/latitude values are successfully located.</td>
</tr>
</tbody>
</table>

Returns

This function returns a callback function that gets the latitude and longitude values of the specified address.

Example
<h3>This is an example of the Map.getLatitudeLongitude function. Click the HTML button labeled "GetLatitude-Longitude" to get the latitude and longitude of Ann Arbor, MI.</h3>

```html
<script>
function getLongitudeLatitude()
{
  ColdFusion.Map.getLatitudeLongitude('201 S. Division St. Suite 500 Ann Arbor, MI 48104', callbackHandler);
}

function callbackHandler(result)
{
  alert("The latitude-longitude of Ann Arbor, MI is: "+result);
}
</script>

<h3>MAP 1</h3>
<cfform name="map01">
  <cfinput type="button" value="GetLatitude-Longitude" name="buttn03" onclick="javascript:getLongitudeLatitude()">
</cfform>
<cfmap name="mapID" centerlatitude= 42 centerlongitude=-87
doubleclickzoom="true"
overview=true
scrollwheelzoom=true
tips="My Map" zoomlevel="4">
</cfmap>

#back to top
ColdFusion.Map.getMapObject

Description

Gets the Google map component. You can manipulate the map using supported Google Map APIs.

Function syntax

```ColdFusion.Map.getMapObject("name")```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the <code>name</code> attribute of the <code>cfmap</code> tag.</td>
</tr>
</tbody>
</table>

Returns

This function returns the Google map component. The map type can be `map`, `satellite`, or `hybrid`.

Example
This is an example of the Map.mapObject function. Click the HTML button labeled "GetMap" to get the map object and set the center to Palo Alto.

```javascript
function getMapObject()
{
    var mapObj = ColdFusion.Map.mapObject('mapID');
    mapObj.setCenter(new GLatLng(37.4419, -122.1419), 13);
}
```

<h3>MAP 1</h3>
```cfcfform name="map01">
    <cfinput type="button" value="getMapObject and setCenter" name="htmlbutton" onclick="javascript:getMapObject()">
</cfform>
```cfmap name="mapID"
    centerAddress='201 S. Division St. Suite 500 Ann Arbor, MI 48104'
    displayscale=true
doubleclickzoom="true"
    overview=true
    scrollwheelzoom=true
tips="My Map"
    zoomlevel="4">
```
ColdFusion.Map.hide

**Description**

If displayed, hides the map.

**Function syntax**

ColdFusion.Map.hide(Id)

**Parameters**

- Id: Name of the map.

**Example**

See example in ColdFusion.Map.show.
ColdFusion.Map.refresh

Description

Reloads the map.

Function syntax

ColdFusion.Map.refresh (Id)

Parameters

- Id: Name of the map.

Usage

If the map is embedded within spry collapsible panels or divs that are hidden on display, that is the map container is displayed while the actual map is hidden, use this function to force the map to display.

Example

```javascript
<script type="text/javascript"
src="/CFIDE/scripts/ajax/spry/includes_minified/SpryCollapsiblePanel.js"></script>
<link type="text/css"
href="/CFIDE/scripts/ajax/spry/widgets/collapsiblepanel/SpryCollapsiblePanel.css"
rel="stylesheet">
<div id="cp" class="CollapsiblePanel" style="width:500px;">
  <div class="CollapsiblePanelTab" tabindex="0">SHOW MAP</div>
  <div class="CollapsiblePanelContent">
    <cfmap
      width="500"
      height="200"
      zoomlevel="12"
      name="mainMap"
      markercolor="333444"
      showscale="false"
      typecontrol="none"
      showcentermarker="true"
      centeraddress="The Key Learning centre, Oxford, UK"
    >
    </cfmap>
  </div>
</div>
<script type="text/javascript">
  var myTabClick = function()
  {
    !cpanel.isOpen() ? cpanel.open() : cpanel.close();
    cpanel.focus();
    ColdFusion.Map.refresh('mainMap');
  }
  var cpanel = new Spry.Widget.CollapsiblePanel("cp", {contentIsOpen:false});
  cpanel.onTabClick = myTabClick;
</script>
```
ColdFusion.Map.setCenter

Description

Sets the center of map to the address that you specify.

Function syntax

```
ColdFusion.Map.setCenter("name", centerConfigObject)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute of the cfmap tag.</td>
</tr>
<tr>
<td>centerConfigObject</td>
<td>The Center Address object. The value of this object can either be the longitude/latitude value or the address property.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example 1
This is an example of the Map.setcenter function using latitude-longitude and address values.

```html
<h3>This is an example of the Map.setcenter function using latitude-longitude and address values</h3>
<script>
    var centerLongLat={
        latitude: 71.094224,
        longitude: 42.339641
    };
    var center={
        address: '345 Park Avenue, san jose, CA 95110-2704, USA'
    };
    function setcenter()
    {
    ColdFusion.Map.setCenter('mapID', centerLongLat);
    }
    function setcenterlatlong()
    {
    ColdFusion.Map.setCenter('mapID', center);
    }
</script>
<h3>MAP 1</h3>
<cfform name="mapID">
    <cfinput type="button" value="setCenter using lattitude-longitude" name="buttn01" onclick="javascript:setcenterlatlong();">
    Click this button to set the center using Latitude and Longitude.
    <cfinput type="button" value="setCenter using Address" name="buttn01" onclick="javascript:setcenter();">
    Click this button to set the center using Address.
</cfform>
<cfmap name="mapID" centerlatitude=71.094224 centerlongitude=42.339641 displayscale=true doubleclickzoom="true" overview=true scrollwheelzoom=true tips="My Map" zoomlevel="4">
</cfmap>
```

#back to top
ColdFusion.Map.setZoomlevel

Description

Sets the zoom level of the map to the new value.

Function syntax

ColdFusion.Map.setZoomlevel("name", zoomLevelValue)

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute of the cfmap tag.</td>
</tr>
<tr>
<td>zoomLevelValue</td>
<td>The value of the integer in the zoomlevel attribute of the cfmap tag.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example
<h3>This is an example of the Map.setzoomlevel function. Click the Set Zoom Level button to set the zoom level to 6.</h3>

```html
<script>
function setZoom(zoomlevel)
{
    ColdFusion.Map.setZoomLevel('mapID', zoomlevel);
}
</script>

<h3>MAP 1</h3>

```html
<cfform name="map01">
<cfinput type="button" value="Set Zoom Level" name="buttn04"
onclick="javascript:setZoom(6)">
</cfform>
```html
<cfmap name="mapID"
centerlatitude=42.094224
centerlongitude=72.339641
displayscale=true
doubleclickzoom="true"
overview=true
scrollwheelzoom=true
tips="My Map"
zoomlevel="4" >
</cfmap>
**ColdFusion.Map.show**

*Description*

Shows the map if it is hidden.

*Function syntax*

ColdFusion.Map.show(Id)

*Parameters*

- **Id**: Name of the map.

*Example*

```<script>
function showMap(mapId)
{
    ColdFusion.Map.show(mapId);
}

function hideMap(mapId)
{
    ColdFusion.Map.hide(mapId);
}
</script>
```

```
<a href="##" id="a1" onclick="return showMap('mainMap')">Show Map</a> | <a href="##" id="a1" onclick="return hideMap('mainMap')">Hide Map</a>
```

```<cfmap
    zoomlevel = "12"
    name = "mainMap"
    showcentermarker = "true"
    centeraddress = "The Key Learning centre, Oxford, UK"
    title="Venue Address"
    hideborder=false
    collapsible=true
    initShow=false/>
```

[#back to top](#)
ColdFusion.MediaPlayer.getPlayer

**Description**

Returns the player object that is used to invoke Strobe media player JavaScript API.

**Function syntax**

```javascript
ColdFusion.MediaPlayer.getPlayer("Name")
```

**See also**


**History**

ColdFusion 10: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specifies the value of the name attribute of the cfmediaplayer tag.</td>
</tr>
</tbody>
</table>

**Example**

See the example for [Dynamic streaming](#back to top)
ColdFusion.Mediaplayer.getType

**Description**

Used to get the current playback type, if Flash or HTML player.

**Function syntax**

```
ColdFusion.MediaPlayer.getType("name")
```

**See also**


**History**

ColdFusion 10: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the <code>name</code> attribute of the <code>cfmediaplayer</code> tag.</td>
</tr>
</tbody>
</table>

**Returns**

This function returns the playback type as Flash or HTML 5.

**Example**
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function getPlayback(player)
{
    alert(ColdFusion.MediaPlayer.getType("player"));
}
</script>
</head>
<body style="background:"
<input name = "Source" value="Type" type="button" onclick="getPlayback('player')">
<div>
<cfmediaplayer
    fullScreenControl="yes"
    name="player"
    width=800
    height=500
    
    <source src="/videos/cathy2_HD.mp4" type="video/mp4" />
    <source src="/videos/gizmo.ogv" type="video/ogg" />
</cfmediaplayer>
</div>
</body>
</html>
ColdFusion.Mediaplayer.logError

Description

Lets you show custom error message on the media player, if an error occurs during playback.

Function syntax

ColdFusion.Mediaplayer.logError("Name", "Error")

See also


History

ColdFusion 10: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specifies the value of the name attribute of the cfmediaplayer tag.</td>
</tr>
<tr>
<td>Error</td>
<td>The custom error message that you want to display.</td>
</tr>
</tbody>
</table>

Example

See the example for ColdFusion.Mediaplayer.getType.

#back to top
ColdFusion.Mediaplayer.resize

Description

Changes the current size of the media player.

Function syntax

```
ColdFusion.Mediaplayer.resize("name", "height", "width")
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute of the cfmediaplayer tag.</td>
</tr>
<tr>
<td>height</td>
<td>The height (in pixels) to set for the media player. The height is 360 pixels, by default.</td>
</tr>
<tr>
<td>width</td>
<td>The width (in pixels) to set for the media player. The width is 480 pixels, by default.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example

In this example, the FLV file is stored in the web root used by the ColdFusion server. Store an FLV file - video.flv in the location _web_root\xyz_.

<h3>This is an example of the Mediaplayer.resize function. Clicking the Resize Mediaplayer button resizes the media player component.</h3>
<script language="JavaScript" type="text/javascript">
function onResize() {
ColdFusion.Mediaplayer.resize('Myvideo', 500, 800);
};
</script>
<br />
<form>
<input type="button" name="resize" value="Resize Mediaplayer" onClick="onResize()">
</form>
<br />
<cfmediaplayer name="Myvideo" source="\xyz\video.flv"
width=500 height=400 align="middle"
quality="high" fullscreencontrol="true"/>
ColdFusion.Mediaplayer.setTitle

Description

Assigns a title for the media player on the top left corner.

Function syntax

ColdFusion.Mediaplayer.setTitle("name", "title")

See also


History

ColdFusion 10: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute of the cfmediaplayer tag.</td>
</tr>
<tr>
<td>title</td>
<td>Specifies the title to be displayed at the top left corner of the media player.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example
<h3>This is an example of the Mediaplayer.setTitle function. Clicking the Resize Mediaplayer button resizes the media player component.</h3>

<script language="JavaScript" type="text/javascript">
    function assignTitle() {
        ColdFusion.Mediaplayer.setTitle('Myvideo', Video);
    }
</script>

<form>
    <input type="button" name="resize" value="Resize Mediaplayer" onClick="assignTitle()">
</form>

<cfmediaplayer name="Myvideo" source="\xyz\video.flv" width=500 height=400 align="middle" quality="high" fullscreencontrol="true"/>
ColdFusion.Mediaplayer.setMute

**Description**

Mutes or unmutes the sound of the media player.

**Function syntax**

```coldfusion
ColdFusion.Mediaplayer.setMute("name", mute)
```

**See also**


**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the <code>name</code> attribute of the <code>cfmediaplayer</code> tag.</td>
</tr>
<tr>
<td>mute</td>
<td>A Boolean value specifying whether to mute the sound (true), or to turn off the mute (false).</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return any value.

**Example**

In this example, the FLV file is stored in the web root used by the ColdFusion server. Store an FLV file - `video.flv` in the location `_web_root\xyz_`. 
This is an example of the Mediaplayer.setmute function. Clicking the Mute button mutes the media player.

```javascript
function onMute() {
    ColdFusion.Mediaplayer.setMute('Myvideo', true);
}
```

<form>
<input type="button" name="mute" value="Mute" onClick="onMute()"/>
</form>
<cfmediaplayer name="Myvideo" source="/xyz/video.flv"
width=500 height=400 align="middle"
quality="high" fullscreencontrol="true"/>
ColdFusion.Mediaplayer.setSource

**Description**

Sets the URL of the FLV file. The URL can point to a location on the ColdFusion server or any other server.

**Function syntax**

```plaintext
ColdFusion.Mediaplayer.setSource("name", newURL)
```

**See also**


**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute of the cfmediaplayer tag.</td>
</tr>
<tr>
<td>newURL</td>
<td>The URL to the FLV file. This can be a URL relative to the current page. You can store the FLV file on the ColdFusion server or any other streaming server.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return any value.

**Example**

In this example, the source of the FLV file is changed from video.flv to newvideo.flv. The media player then plays the newvideo.flv file.
<h3>This is an example of the Mediaplayer.setsource function. Clicking the Set Source button changes the video playing in the media player.</h3>

```javascript
function setSource()
{
    ColdFusion.Mediaplayer.setSource('Myvideo', '/xyz/newvideo.flv');
}
</script>
<br>
<form>
<input type="button" name="setSource" value="Set Source" onClick="setSource()">
</form>
<cfmediaplayer name="Myvideo" source="/xyz/video.flv" width=500 height=400 align="middle" quality="high" fullscreencontrol="true"/>
ColdFusion.Mediaplayer.setVolume

Description

Sets the volume of sound of the media player.

Function syntax

ColdFusion.Mediaplayer.setVolume("name", "volume")

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute of the cfmediaplayer tag.</td>
</tr>
<tr>
<td>volume</td>
<td>The value of the volume. It can take any value from 0.0 through 1.0.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example

In this example, the FLV file is stored in the web root used by the ColdFusion server. Store an FLV file - video.flv in the location _web_root\xyz_.

© 2014 Adobe Systems Incorporated. All rights reserved.
<h3>This is an example of the Mediaplayer.setvolume function. Clicking the Set Volume button increases the volume of the media player.</h3>
<script language="JavaScript" type="text/javascript">
    function setVol() {
        ColdFusion.Mediaplayer.setVolume('Myvideo', 1.0);
    }
</script>
<form>
    <input type="button" name="setVolume" value="Set Volume" onClick="setVol()">
</form>
<cfmediaplayer name="Myvideo" source="\xyz\video.flv" width=500 height=400 align="middle" quality="high" fullscreencontrol="true"/>
ColdFusion.Mediaplayer.startPlay

Description

Plays the FLV file.

Function syntax

ColdFusion.Mediaplayer.startPlay("name")

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the <code>name</code> attribute of the <code>&lt;cfmediaplayer&gt;</code> tag.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example

In this example, the FLV file is stored in the web root used by the ColdFusion server. Store an FLV file - video.flv in the location `_web_root\xyz_`.

```html
<h3>This is an example of the Mediaplayer.startplay function. Clicking the Start Play button plays the video.</h3>
<script language="JavaScript" type="text/javascript">
  function onStart() {
    ColdFusion.Mediaplayer.startPlay('Myvideo');
  }
</script>
<form>
  <input type="button" name="startplay" value="Start Play" onClick="onStart()">
</form>
<cfmediaplayer name="Myvideo" source="/xyz/video.flv" width=500 height=400 align="middle" quality="high"fullscreencontrol="true"/>
```
**ColdFusion.Mediaplayer.stopPlay**

**Description**

Stops playing the FLV file.

**Function syntax**

```javascript
ColdFusion.Mediaplayer.stopPlay("name")
```

**See also**


**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute of the cfmediaplayer tag.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return any value.

**Example**

In this example, the FLV file is stored in the web root used by the ColdFusion server. Store an FLV file - video.flv in the location _web_root\xyz_.

```html
<h3>This is an example of the Mediaplayer.stopplay function. Clicking the Stop Play button stops playing the video.</h3>
<script language="JavaScript" type="text/javascript">
function onStop()
{
ColdFusion.Media player.stopPlay('Myvideo' );
}
</script>
<form>
<input type="button" name="stopPlay" value="Stop Play" onClick="onStop()"/>
</form>
<cfmediaplayer name="Myvideo" source="/xyz/video.flv" width=500 height=400 align="middle" quality="high" fullscreencontrol="true"/>
```
ColdFusion.MessageBox.create

Description

Creates a ColdFusion message box. This function is equivalent to the cfmessagebox tag.

Function syntax

ColdFusion.MessageBox.create(name, type, title, message, callbackhandler [, configuration])

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the message box. This attribute is required to show and update the message box. The message box name must be unique on the page.</td>
</tr>
</tbody>
</table>
| type            | The control type. Must be one of the following:  
  - alert - A message with a single OK button.  
  - confirm - A message box with two buttons YES and NO or three buttons YES, NO, and CANCEL.  
  - prompt - A message box with a single-line or multiline text input area and OK and CANCEL buttons. |
| title           | The text to display on the message box title bar. |
| message         | The text to display inside the message box. |
| callbackhandler | The function that the control calls when a user clicks one of the buttons. |
| configuration   | An object containing message box configuration parameters. For details, see "Usage". |

Returns
This function does not return a value.

Usage

This function is equivalent to the `cfmessagebox` tag. If you do not also use a `cfmessagebox` tag on a page that calls this function, specify a `cfajaximport` tag on the page and specify `cfmessagebox` in the `tags` attribute. Doing so ensures that the page includes the necessary JavaScript to create the message box. For example, use the following line if you do not have to import the JavaScript for any other ColdFusion Ajax features:

```xml
<cfajaximport tags="cfmessagebox">
```

The `configuration` parameter defines the message box characteristics; it can have any or all the following entries:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodystyle</td>
<td></td>
<td>A CSS style specification for the body of the message box. Generally, you use this attribute to set color and font styles.</td>
</tr>
<tr>
<td>buttontype</td>
<td>yesno</td>
<td>Applies to the control type <code>confirm</code>. The buttons to display on the message box:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yesno: displays the buttons Yes and No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yesnocancel: displays the buttons Yes, No, and Cancel</td>
</tr>
<tr>
<td>icon</td>
<td></td>
<td>Specifies the following CSS classes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• error: Provides the error icon. You can use this icon when displaying error messages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• info: Provides the info icon. You can use this icon when displaying any information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• question: Provides the question icon. You can use this icon in a confirmation message box that prompts a user response.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• warning: Provides the warning icon. You can use this icon when displaying a warning message.</td>
</tr>
<tr>
<td>labelcancel</td>
<td>Cancel</td>
<td>The text to put on the cancel button of a prompt message box.</td>
</tr>
<tr>
<td>labelok</td>
<td>OK</td>
<td>The text to put on an alert button and prompt message box OK button.</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>labelno</td>
<td>No</td>
<td>The text to put on the button used for a negative response in a confirm message box.</td>
</tr>
<tr>
<td>labelyes</td>
<td>Yes</td>
<td>The text to put on the button used for a positive response a confirm message box.</td>
</tr>
<tr>
<td>modal</td>
<td>yes</td>
<td>A Boolean value that specifies if the message box is a modal window:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• no</td>
</tr>
<tr>
<td>multiline</td>
<td>false</td>
<td>Valid only for prompt type message boxes. A Boolean value specifying whether the prompt input text box has a single or multiple lines for text input.</td>
</tr>
<tr>
<td>width</td>
<td></td>
<td>Width of the message box in pixels.</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>The X (horizontal) coordinate of the upper-left corner of the message box. ColdFusion ignores this attribute if you do not set the y attribute.</td>
</tr>
<tr>
<td>y</td>
<td></td>
<td>The Y (vertical) coordinate of the upper-left corner of the message box. ColdFusion ignores this attribute if you do not set the x attribute.</td>
</tr>
</tbody>
</table>

⚠️ Note

Entry names in the configuration object must be all-lowercase.

Example

The following minimal CFML application creates a message box of type confirmation.
<cfajaximport tags="cfmessagebox">
<cfform name="test">
<cfinput type="button" name="x" value="Create Message Box"
    onClick="ColdFusion.MessageBox.create('Messagebox1', 'confirm', 'Confirm',
        'Do you want to save the file?',
        onfinish, {width:200, modal: false})">
</cfform>
<script language="JavaScript" type="text/javascript">
    function onfinish()
    {
        alert('Button clicked');
    }
</script>
ColdFusion.MessageBox.show

Description

Used to display a ColdFusion message box.

Function syntax

ColdFusion.MessageBox.show(name)

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the message box that you want to display.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

You can create a messagebox using the cfmessagebox tag or the JavaScript function ColdFusion.MessageBox.create. But to show it, you must use this function.

Example

```html
<cfajaximport tags="cfmessagebox,cfform">
<cfform name="ajax">
  <br><input type="button" name="showMessageBox" value="Show Message Box"
onClick="ColdFusion.MessageBox.create('mb','Alert','ALERT','Sample Alert!');ColdFusion.MessageBox.show('mb');"/>
</cfform>
```

#back to top
ColdFusion.MessageBox.getMessageBoxObject

Description

Gets the underlying Ext JS - JavaScript Library object for the specified HTML cfmessagebox control.

Function syntax

```
ColdFusion.MessageBox.getMessageBoxObject(name)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the message box object.</td>
</tr>
</tbody>
</table>

Returns

A JavaScript object.

Usage

Use this function to get the JavaScript object that contains all the defined properties. ----

#back to top
ColdFusion.MessageBox.isMessageBoxDefined

Description

Checks if a message box is defined.

Function syntax

```
ColdFusion.MessageBox.isMessageBoxDefined(name)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the message box object.</td>
</tr>
</tbody>
</table>

Returns

A Boolean value, that is, true or false.

Usage

Use this function to check if the message box is defined for a specific name.

#back to top
ColdFusion.MessageBox.update

Description

Updates the ColdFusion message box properties. This JavaScript function lets you update all the message box properties except `name` and `type`.

Function syntax

```
ColdFusion.MessageBox.update(name, configuration)
```

See also

ColdFusion.MessageBox.create, ColdFusion.MessageBox.getMessageBoxObject, ColdFusion.MessageBox.isMessageBoxDefined, ColdFusion.MessageBox.updateMessage, ColdFusion.MessageBox.updateTitle

History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the message box. This attribute is required to show and update the message box. The message box name must be unique on the page.</td>
</tr>
<tr>
<td>configuration</td>
<td>An object containing message box configuration parameters. For details, see &quot;Usage&quot;.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function is used to update a set of message box properties. For example, you can update the width, message, and title of the message box using the `configuration` parameter. The `configuration` parameter defines the message box characteristics; it can have any or all the following entries:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodystyle</td>
<td>A CSS style specification for the body of the message box. Generally, you use this attribute to set color and font styles.</td>
</tr>
</tbody>
</table>
| **buttontype**         | Applies to the control type - confirm. The buttons to display on the message box:  
|                        | • yesno: displays the buttons Yes and No  
|                        | • yesnocancel: displays the buttons Yes, No, and Cancel  |
| **callbackhandler**    | The function that the control calls when a user clicks one of the buttons. For more information see Usage. |
| **icon**               | Specifies the following CSS classes:  
|                        | • error: Provides the error icon. You can use this icon when displaying error messages.  
|                        | • info: Provides the info icon. You can use this icon when displaying any information.  
|                        | • question: Provides the question icon. You can use this icon in a confirmation message box that prompts a user response.  
|                        | • warning: Provides the warning icon. You can use this icon when displaying a warning message. |
| **labelcancel**        | The text to put on the cancel button of a prompt message box. |
| **labelok**            | The text to put on an alert button and prompt message box OK button. |
| **labelno**            | The text to put on the button used for a negative response in a confirm message box. |
| **labelyes**           | The text to put on the button used for a positive response in a confirm message box. |
| **modal**              | A Boolean value that specifies if the message box is a modal window:  
|                        | • yes  
<p>|                        | • no  |
| <strong>message</strong>            | The text to display inside the message box. |
| <strong>multiline</strong>          | Valid only for prompt type message boxes. A Boolean value specifying whether the prompt input text box has a single or multiple lines for text input. |
| <strong>title</strong>              | The title for the message box. If you do not specify a title, ColdFusion assigns the control type value as the default title. |</p>
<table>
<thead>
<tr>
<th>width</th>
<th>Width of the message box in pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The X (horizontal) coordinate of the upper-left corner of the message box. ColdFusion ignores this attribute if you do not set the y attribute.</td>
</tr>
<tr>
<td>y</td>
<td>The Y (vertical) coordinate of the upper-left corner of the message box. ColdFusion ignores this attribute if you do not set the x attribute.</td>
</tr>
</tbody>
</table>

⚠️ Note

Entry names in the configuration object must be all-lowercase.

Example

The following minimal CFML application updates a message box.

```cfml
<cfajaximport tags="cfmessagebox">
<cfform name="test">
  <cfinput type="button" name="x" value="Update Message Box" onClick="ColdFusion.MessageBox.update('Messagebox1', {width:400, modal:false, x:200, y:300, labelyes:'yes'})">
</cfform>
```

#back to top
ColdFusion.MessageBox.updateMessage

**Description**

Updates the message property of the ColdFusion message box component.

**Function syntax**

```
ColdFusion.MessageBox.updateMessage(name, newmessage)
```

**See also**


**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the message box object.</td>
</tr>
<tr>
<td>newmessage</td>
<td>Overwrites an existing message.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return a value.

**Usage**

Use this function to update or change the message property of the message box. ----

#back to top
ColdFusion.MessageBox.updateTitle

Description

Updates the title property of the ColdFusion message box component.

Function syntax

```coldfusion
ColdFusion.MessageBox.updateTitle(name, newtitle)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the message box object.</td>
</tr>
<tr>
<td>newmessage</td>
<td>Overwrites an existing title.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

Use this function to update or change the title property of the message box.

#back to top
ColdFusion.navigate

Description

Displays the output of a link target in an Ajax cfdiv, cflayoutarea, cfpod, or cfwindow container. When the browser follows a link that is populated by this function, the link does not replace the current page. Instead, it populates the control specified by the container attribute.

Function syntax

ColdFusion.navigate(URL [, container, callbackhandler, errorHandler, httpMethod, formId])

See also

AjaxLink, cfajaximport, ColdFusion.Ajax.submitForm, Control container contents in Controlling Ajax user interface layout in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>The URL of the link.</td>
</tr>
<tr>
<td>container</td>
<td>The name attribute value of the control in which to display the link output. The control must be a container control such as cfdiv, cflayoutarea, cfpod, or cfwindow. If you omit this argument, the link is treated as a normal URL and the entire page is refreshed.</td>
</tr>
<tr>
<td>callbackhandler</td>
<td>The name of a JavaScript function to call after the target has been displayed.</td>
</tr>
<tr>
<td>errorHandler</td>
<td>The name of a JavaScript function to call if an error occurs when this function executes. The function can take two parameters: an HTTP error code, and an error message.</td>
</tr>
<tr>
<td>formId</td>
<td>The ID or name attribute of a form to submit to the URL.</td>
</tr>
<tr>
<td>httpMethod</td>
<td>The HTTP method to use when navigating to the URL:</td>
</tr>
<tr>
<td></td>
<td>• GET (the default)</td>
</tr>
<tr>
<td></td>
<td>• POST</td>
</tr>
</tbody>
</table>
Returns

This function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, you must use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function. The `callbackhandler` parameter can be useful for changing the display after the contents has been displayed. For example, before you make the `ColdFusion.navigate` call you might make a pod's title bar italic to indicate loading; you could then use the `callbackhandler` function to switch it back to normal or make it bold once `navigate` completes. Similarly, you could use a `callbackhandler` to update the page number in a book reader. The `FormID` attribute lets you specify a form to submit to the specified URL. You can use the `ColdFusion.Navigate` function with this attribute to submit form data asynchronously from outside the form, for example, when the user clicks a menu item, and to direct the returned results to a specific container control.

Example

When the user clicks the link in window 1, the `ColdFusion.navigate` function replaces the text in window 2 with the contents of `windowsrc.cfm`, and then calls the myCallback callback handler, which changes the `innerHTML` of the callback div region. The main application page looks as follows:

```html
<html>
<head>
<!--- The Callback handler puts text in the window.cfm callback div. --->
<script language="javascript">
  var mycallBack = function(){
    document.getElementById("callback").innerHTML = "<br><br><b>This is printed by the callback handler.</b>";
  }
<!--- The error handler pops an alert with the error code and message. --->
  var myerrorhandler = function(errorCode,errorMessage){
    alert("[In Error Handler]" + "\n\n" + "Error Code: " + errorCode + "\n\n" + "Error Message: " + errorMessage);
  }
</script>
</head>

<body>
<cfwindow name="w1" title="CF Window 1" initShow=true x=10 y=10 width="200">
  This is a cfwindow control.<br><br>
  <a href="javascript:ColdFusion.navigate('windowsource.cfm','w2',mycallBack,myerrorhandler);">Click</a> to navigate Window 2</a>
</cfwindow>

<cfwindow name="w2" title="CF Window 2" initShow=true x=250 y=10 width="200">
  This is a second cfwindow control.
</cfwindow>
</body>
</html>
```
The windowsource.cfm page looks as follows:

- This is markup from "windowsource.cfm"
- <!-- The callback handler puts its output in the following div block. -->
- <div id="callback"></div>
ColdFusion.ProgressBar.getProgressBarObject

Description

Gets the progress bar object.

Function syntax

ColdFusion.ProgressBar.getProgressBarObject(name)

See also

ColdFusion.ProgressBar.start, ColdFusion.ProgressBar.stop

History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the progress bar object.</td>
</tr>
</tbody>
</table>

Returns

This function returns the underlying Ext JavaScript progress bar object.

Usage

You call this function to get the progress bar object. ----

#back to top
ColdFusion.ProgressBar.hide

**Description**

Hides the progress bar if it is displayed.

**Function syntax**

```
ColdFusion.ProgressBar.hide(progressBarId)
```

**See also**

ColdFusion.ProgressBar.show, ColdFusion.ProgressBar.update

**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>progressBarId</td>
<td>Name of the progress bar object. This must be a valid ColdFusion identifier.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return a value.

**Usage**

This function lets you hide the progress bar if it is displayed. You display the progress bar using the function ColdFusion.ProgressBar.show or by setting the attribute autodisplay to true in the tag cfprogressbar. The following example shows how to use this function:

```
<cfprogressbar name="pBar" autodisplay="true">
<cfinput type="button"
    onClick="ColdFusion.ProgressBar.hide('pBar')">
```

#back to top
ColdFusion.ProgressBar.reset

Description

Resets the progress status and messages.

Function syntax

```
ColdFusion.ProgressBar.reset(progressBarId)
```

See also

ColdFusion.ProgressBar.start, ColdFusion.ProgressBar.stop

History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>progressBarId</td>
<td>Name of the progress bar object. This must be a valid ColdFusion identifier.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

Resets the progress on a progress bar as shown in the following example:

```
<cfform>
  <cfprogressbar name="pBar" width="500"/>
  <cfinput type="button" name="pBtn" onClick="ColdFusion.ProgressBar.reset('pBar')">
</cfform>
```
**ColdFusion.ProgressBar.show**

**Description**

Shows the progress bar if it is hidden.

**Function syntax**

```
ColdFusion.ProgressBar.show(progressBarId)
```

**See also**

[ColdFusion.ProgressBar.hide](#), [ColdFusion.ProgressBar.update](#)

**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>progressBarId</td>
<td>Name of the progress bar object. This must be a valid ColdFusion identifier.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return a value.

**Usage**

This function lets you display the progress bar if it is hidden. You hide the progress bar using the function [ColdFusion.ProgressBar.hide](#) or by setting the attribute `autodisplay` to `false` in the `cfprogressbar` tag. The following example shows how to use this function:

```
<cfprogressbar name="pBar" autodisplay="false">
<cfinput type="button"
   onClick="ColdFusion.ProgressBar.show('pBar')">
```

[#back to top]
ColdFusion.ProgressBar.start

Description

Starts the underlying Ext JS - JavaScript Library progress bar object.

Function syntax

```
ColdFusion.ProgressBar.start(name)
```

See also

ColdFusion.ProgressBar.getProgressBarObject, ColdFusion.ProgressBar.stop

History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the progress bar object.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

The `cfprogressbar` tag creates the HTML markup; at runtime, you use this function to initiate the progress bar object. On initialization, ColdFusion calls the underlying CFC, which is defined using a bind expression, at specified intervals. The CFC returns the progress status, which is passed to the underlying Ext progress bar object to update the progress bar value. The progress status object that the CFC returns must have STATUS and MESSAGE properties. The STATUS property has a numeric value from 0.0 to 1.0.

#back to top
ColdFusion.ProgressBar.stop

Description

Stops the underlying progress bar object that is running.

Function syntax

```
ColdFusion.ProgressBar.stop(name, calloncomplete)
```

See also

ColdFusion.ProgressBar.getProgressBarObject, ColdFusion.ProgressBar.start

History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the progress bar object.</td>
</tr>
<tr>
<td>calloncomplete</td>
<td>A Boolean value that specifies whether to call oncomplete function:</td>
</tr>
<tr>
<td></td>
<td>• true</td>
</tr>
<tr>
<td></td>
<td>• false</td>
</tr>
<tr>
<td></td>
<td>The default value is true.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

You call this function to stop the progress bar object.

#back to top
ColdFusion.ProgressBar.update

**Description**

Updates attribute values.

**Function syntax**

```javascript
ColdFusion.ProgressBar.update(progressBarId, config)
```

**See also**

`ColdFusion.ProgressBar.hide`, `ColdFusion.ProgressBar.show`

**History**

ColdFusion 9: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>progressBarId</td>
<td>Name of the progress bar object. This must be a valid ColdFusion identifier.</td>
</tr>
<tr>
<td>configObject</td>
<td>An object containing configuration parameters, interval, duration, and oncomplete.</td>
</tr>
</tbody>
</table>

**Returns**

This function does not return a value.

**Usage**

Updates any of the attribute values `duration`, `interval`, or `oncomplete` as shown in the following example:

```cfml
<cfprogressbar name="pBar" autodisplay="false">
<cfinput type="button" onClick="ColdFusion.ProgressBar.update('pBar', {interval:6000, duration:10000, oncomplete:newoncomplete})">
```

#back to top
ColdFusion.ProgressBar.updatestatus

Description

Lets you manually update the status and message of the progress bar.

Function syntax

```
ColdFusion.ProgressBar.updatestatus(progressBarId, status, message)
```

See also

ColdFusion.ProgressBar.update, ColdFusion.ProgressBar.reset

History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>progressBarId</td>
<td>Name of the progress bar object. This must be a valid ColdFusion identifier.</td>
</tr>
<tr>
<td>message</td>
<td>Text to be displayed in the progress bar.</td>
</tr>
<tr>
<td>status</td>
<td>Progress status.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function helps you to manually set the message and status in the progress bar. For instance, in the case of file upload, you can manually control the progress using this function. The following snippet illustrates how to use the function:

```
<cfform>
  <cfprogressbar name="pBar" width="500"/>
  <cfinput type="button" name="pBtn" onClick="ColdFusion.ProgressBar.updateStatus('pBar', 0.5, '50%')">
</cfform>
```

#back to top
ColdFusion.RichText.getEditorObject

Description

Returns the FCKEditor instance that underlies an existing textarea control's rich text editor.

Function syntax

```
object = ColdFusion.RichText.getEditorObject(textareaName)
```

See also

ColdFusion.ProgressBar.update, ColdFusion.ProgressBar.reset

History

ColdFusion 8 update 1: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>textareaName</td>
<td>The value of the name attribute of the textarea tag that specifies a rich text editor.</td>
</tr>
</tbody>
</table>

Returns

An object.

Usage

This function returns an object that provides access to the operations of the internal FCKEditor.

#back to top
ColdFusion.RichText.onComplete

Description

When a rich text editor instance completes initializing, ColdFusion calls this function and passes it the FCKEditor instance that underlies the rich text editor.

Function syntax

ColdFusion.RichText.onComplete(editorInstance)

See also

ColdFusion.ProgressBar.update, ColdFusion.ProgressBar.reset

History

ColdFusion 8 update 1: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>editorInstance</td>
<td>the FCKEditor instance that underlies the rich text editor.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

When a rich text editor instance completes initializing, ColdFusion calls this function and passes it the FCKEditor instance that underlies the rich text editor. The function can then perform any needed post-initialization operations.

Examples
<html>
<head>
<script type="text/javascript">
//User defines this function to get control once the rich text editor
//is done with its initialization
ColdFusion.RichText.OnComplete = function(editorInstance)
{
  alert("on complete called");
  alert("editor instance = " + editorInstance));
}
</script>
</head>
<body>
<cfform>
  <cftextarea richtext = true name="richtext1">
  </cftextarea>
</cfform>
</body>
</html>
ColdFusion.setGlobalErrorHandler

Description

Specifies a function that gets called, in place of the ColdFusion Ajax default error handler, if an error occurs when using a ColdFusion Ajax feature.

Function syntax

```
ColdFusion.setGlobalErrorHandler(functionName)
```

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>functionName</td>
<td>The name of the JavaScript function to execute when there is an error in ColdFusion Ajax code, such as a binding error. This function must take a single argument, the error message string.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

If the page that calls this function does not have any ColdFusion AJAX-based controls, use a `cfajaximport` tag on the page to ensure that the page includes the JavaScript definition for this function. The global error handler displays information about errors that occur in ColdFusion Ajax features. The default global error handler displays an alert with the error message. You can use this function to create a custom global error handler, for example, to display a custom error window with additional information about your application.

#back to top
ColdFusion.Slider.disable

Description

Disables the slider control.

Function syntax

```
ColdFusion.Slider.disable(name)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute in the cfslider tag.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example

```
<h3>This is an example of the Slider.disable function. Click the Disable Slider button to disable the slider.</h3>
<script language="JavaScript" type="text/javascript">
function disable()
{ 
  ColdFusion.Slider.disable('sliderID');
}
</script>
<br>
<cfform name="form01">
<br/>
<cfslider name="sliderID" format="HTML"
vertical="false" width="350"
value="100" min="0" max="200" increment="10" tip="true"/>
<cfinput type="button" name="htmlbutton"
value="Disable Slider" onclick="disable()">
</cfform>
```
ColdFusion.Slider.enable

Description

Enables the slider control.

Function syntax

ColdFusion.Slider.enable(name)

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute in the cfslider tag.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example
<h3>This is an example of the Slider.enable function. Click the Enable Slider button to enable the slider.</h3>

```
function onload()
{
    ColdFusion.Slider.disable('sliderID');
}
function enable()
{
    ColdFusion.Slider.enable('sliderID');
}
```

```
<cfform name="form01">
<br/>
<cfslider name="sliderID" format="HTML"
    body onLoad="onload()"
    vertical="false" width="350"
    value="100" min="0" max="200"
    increment="10" tip="true"/>
<cfinput type="button" name="htmlbutton"
    value="Enable Slider" onclick="enable()"/>
</cfform>
```
ColdFusion.Slider.getSliderObject

Description

Gets the underlying Ext JavaScript slider control.

Function syntax

ColdFusion.Slider.getSliderObject(name)

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute in the cfslider tag.</td>
</tr>
</tbody>
</table>

Returns

This function returns the underlying Ext JavaScript slider control.

#back to top
ColdFusion.Slider.getValue

Description

Gets the numeric value of the slider control.

Function syntax

```
ColdFusion.Slider.getValue(name)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute in the cfslider tag.</td>
</tr>
</tbody>
</table>

Returns

This function returns a numeric value.

Example

```
<h3>This is an example of the Slider.getValue function. Click the Get Value button to get the value of the slider.</h3>
<script language="JavaScript" type="text/javascript">
function getvalue(){
    alert("The slider is currently at: "+ColdFusion.Slider.getValue('sliderID'));
}
</script>
<br/>
<form name="form01">
<br/>
<cfslider name="sliderID" format="HTML" vertical="false" width="350" value="100" min="0" max="200" increment="10" tip="true"/>
<cfinput type="button" name="htmlbutton" value="Get Value" onclick="getvalue()">
</form>
```
ColdFusion.Slider.hide

Description

Hides the slider control.

Function syntax

ColdFusion.Slider.hide(name)

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute in the cfslider tag.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example

```html
<h3>This is an example of the Slider.hide function. Click the Hide button to hide the slider.</h3>
<script language="JavaScript" type="text/javascript">
function hide(){
    ColdFusion.Slider.hide('sliderID');
}
</script>
<br>
<cfform name="form01">
<br>
<cfslider name="sliderID" format="HTML" vertical="false" width="350" value="100" min="0" max="200" increment="10" tip="true"/>
<cfinput type="button" name="htmlbutton" value="Hide" onclick="hide()" value="" onclick="hide()"/>
</cfform>
```
ColdFusion.Slider.setValue

*Description*

Sets the numeric value of the slider control.

*Function syntax*

```
ColdFusion.Slider.setValue(name, newValue)
```

*See also*


*History*

ColdFusion 9: Added this function

*Parameters*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute of the cfmediaplayer tag.</td>
</tr>
<tr>
<td>newValue</td>
<td>The numeric value of the slider control.</td>
</tr>
</tbody>
</table>

*Returns*

This function does not return any value.

*Example*
<h3>This is an example of the Slider.setValue function. Click the Set Value button to set the pointer to a new value, 150.</h3>

```javascript
function setValue()
    ColdFusion.Slider.setValue('sliderID', '150');
}
</script>

<br>
<cfform name="form01">
<br/>
<cfslider name="sliderID" format="HTML"
vertical="false" width="350"
value="100" min="0" max="200"
increment="10" tip="true"/>
<cfinput type="button" name="htmlbutton"
value="Set Value" onclick="setValue()">
</cfform>
ColdFusion.Slider.show

Description
Shows the slider control.

Function syntax

```
ColdFusion.Slider.show(name)
```

See also


History

ColdFusion 9: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the value of the name attribute in the cfslider tag.</td>
</tr>
</tbody>
</table>

Returns

This function does not return any value.

Example
This is an example of the Slider.show function. Click the Show Slider button to show the slider.

```javascript
function onload(){
    ColdFusion.Slider.hide('sliderID');
}
function show(){
    ColdFusion.Slider.show('sliderID');
}
</script>
<br>
<cfform name="form01">
<br/>
<cfslider name="sliderID" format="HTML"
vertical="false" width="350"
value="100" min="0" max="200"
increment="10" tip="true"/>
<cfinput type="button" name="htmlbutton"
value="Show Slider" onclick="show()"/>
</cfform>
ColdFusion.Tree.getTreeObject

Description

Gets the underlying object for the specified HTML tree.

Function syntax

ColdFusion.Tree.getTreeObject(name)

See also

cf.tree, cfajaximport, ColdFusion.Tree.refresh, Using HTML trees in Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute of the cftree tag for which you want the object.</td>
</tr>
</tbody>
</table>

Returns

An object of type YAHOO.widget.TreeView.

Usage

Use this function to get the Yahoo User Interface Library YAHOO.widget.TreeView object that underlies the HTML cftree control. You can then use the raw object to modify the displayed tree. For documentation on the objects and how to manage them, see the [Yahoo toolkit documentation](http://developer.yahoo.com/yui/).
ColdFusion.Tree.refresh

**Description**

Refreshes an HTML tree and updates it with the latest values of all items.

**Function syntax**

```coldfusion
ColdFusion.Tree.refresh(name)
```

**See also**

cftree, cfajaximport, ColdFusion.Tree.getTreeObject, Using HTML trees in Using Ajax form controls and features in the Developing ColdFusion Applications

**History**

ColdFusion 8: Added this function

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute of the cftree tag for which you want the object.</td>
</tr>
</tbody>
</table>

**Returns**

An object of type YAHOO.widget.TreeView.

**Usage**

Use this function to manually update the tree. If you populate the tree by using a bind expression, the refresh call causes the bind expression to be re-evaluated and repopulates the tree root nodes. Use this function any time you must get the latest data from the server independent of an event that triggers the cftree bind expression, for example, use this function to periodically refresh a file/folder tree to represent the status of the server.

#back to top
ColdFusion.Window.create

Description

Creates a ColdFusion pop-up window. This function is equivalent to the `cfwindow` tag.

Function syntax

```plaintext
ColdFusion.Window.create(name, title, URL [, configuration])
```

See also

`cfwindow`, `ColdFusion.Window.getWindowObject`, `ColdFusion.Window.hide`, `ColdFusion.Window.onHide`, `ColdFusion.Window.onShow`, `ColdFusion.Window.show`, `ColdFusion.Tree.getTreeObject`, `Using pop-up windows` in Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the window. This attribute is required to interact with the window, including to dynamically show or hide it. If a window with the specified name exists, the function shows that window, and ignores the remaining parameters; otherwise, the name must be unique on the page.</td>
</tr>
<tr>
<td>title</td>
<td>The text to display on the window title bar. You can use HTML mark-up to control the title appearance.</td>
</tr>
<tr>
<td>URL</td>
<td>The URL from which to get the window body contents. This attribute can use URL parameters to pass data to the page. ColdFusion uses standard page path resolution rules to locate the page. <strong>Note:</strong> If the page specified in this attribute contains tags that use ColdFusion Ajax features, such as the <code>cform</code>, <code>cfgrid</code>, and <code>cfpod</code> tags, identify the tags in a <code>cfajaximport</code> tag on the page that includes this function. For more information, see <code>cfajaximport</code>.</td>
</tr>
<tr>
<td>configuration</td>
<td>An object containing window configuration parameters. For details, see &quot;Usage&quot;.</td>
</tr>
<tr>
<td>refreshonshow</td>
<td>The default value is <code>false</code>.</td>
</tr>
</tbody>
</table>

Returns
This function does not return a value.

**Usage**

This function is equivalent to the `cfwindow` tag. If you do not also use a `cfwindow` tag on a page that calls this function, specify a `cfajaximport` tag on the page and specify `cfwindow` in the `tags` attribute. Doing so ensures that the page includes the necessary JavaScript to create the window. For example, use the following line if you do not have to import the JavaScript for any other ColdFusion Ajax features:

```
<cfajaximport tags="cfwindow">
```

The `configuration` parameter defines the window characteristics; it can have any or all the following entries:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callbackhandler</td>
<td></td>
<td>A function that is called when the window body loads. This function must not take any arguments.</td>
</tr>
</tbody>
</table>
| center | false | A Boolean value that specifies whether to center the window over the browser window.  
  • If `true`, ColdFusion ignores the `x` and `y` attribute values.  
  • If `false`, and you do not specify `x` and `y` attributes, ColdFusion centers the window. |
| closable | true | A Boolean value that specifies whether the user can close the window. If `true`, the window has an X close icon. |
| draggable | true | A Boolean value that specifies whether the user can drag the window. To drag the window, click the mouse on the title bar and hold the button down while dragging. If the window does not have a title, users cannot drag it. |
| **errorHandler** | | A function that is called if an error occurs in loading the window body. This function must take two arguments:
- The HTTP status code, or -1 if the error is not an HTTP error
- An error message |
<p>| <strong>height</strong> | 300 | Height of the window in pixels. If you specify a value greater than the available space, the window occupies the available space and the resize handles do not appear. |
| <strong>initshow</strong> | false | A Boolean value that specifies whether to display the window when the containing page first displays. If this value is false, use the <code>ColdFusion.Window.show</code> JavaScript function to display the window. |
| <strong>minheight</strong> | 0 | The minimum height, in pixels, to which users can resize the window. Specifying this parameter and a <code>resizable=&quot;false&quot;</code> parameter causes an error. |
| <strong>minwidth</strong> | 0 | The minimum width, in pixels, to which users can resize the window. Specifying this parameter and a <code>resizable=&quot;false&quot;</code> parameter causes an error. |
| <strong>modal</strong> | false | A Boolean value that specifies whether the window is modal, that is, whether the user can interact with the main window while this window is displaying. If true, the user cannot interact with the main window. |
| <strong>resizable</strong> | true | A Boolean value that specifies whether the user can resize the window. |</p>
<table>
<thead>
<tr>
<th>width</th>
<th>500</th>
<th>Width of the window in pixels. If you specify a value greater than the available space, the window occupies the available space and the resize handles do not appear.</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td>The X (horizontal) coordinate of the upper-left corner of the window, relative to the browser window. ColdFusion ignores this attribute if the <code>center</code> attribute value is <code>true</code>, and if you do not set the <code>y</code> attribute value.</td>
</tr>
<tr>
<td>y</td>
<td></td>
<td>The Y (vertical) coordinate of the upper-left corner of the window, relative to the browser window. ColdFusion ignores this attribute if the <code>center</code> attribute value is <code>true</code>, and if you do not set the <code>x</code> attribute value.</td>
</tr>
</tbody>
</table>

⚠️ **Note**

Entry names in the configuration object must be all-lowercase.

**Example**

The following minimal CFML application creates a window and gets the window contents from the hello1.cfm file.

```cfml
<cfajaximport tags="cfwindow">
<cfform name="test">
<cfinput type="button" name="x" value="Create Window" onClick="ColdFusion.Window.create('Window1', 'This is a CF window', 'http://localhost:8500/My_stuff/AjaxUI/Book/hello1.cfm', {x:100,y:100,height:300,width:400,modal:false,closable:false,draggable:true,resizable:true,center:true,initshow:true, minheight:200,minwidth:200 })">
</cfform>

The hello1.cfm file can be as simple as the following line:

```
ColdFusion.Window.destroy

Description

Destroys a window instance

Function syntax

```
ColdFusion.Window.destroy(windowName[, destroyElement])
```

See also

cfwindow, ColdFusion.Window.create, ColdFusion.Window.getWindowObject, ColdFusion.Window.onHide, ColdFusion.Window.onShow, ColdFusion.Window.show, ColdFusion.Tree.getTreeObject, Using pop-up windows in Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8 update 1: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>windowName</td>
<td>The name of the window to destroy</td>
</tr>
<tr>
<td>destroyElement</td>
<td>A Boolean value specifying whether the HTML element associated with the window should also be destroyed. The default value is false.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

Use this function to destroy a window instance. ----

#back to top
ColdFusion.Window.getWindowObject

Description

Gets the underlying object for the specified window.

Function syntax

```
ColdFusion.Window.getWindowObject(name)
```

See also

cfwindow, ColdFusion.Window.create, ColdFusion.Window.onHide, ColdFusion.Window.onShow, ColdFusion.Window.show, ColdFusion.Tree.getTreeObject. Using pop-up windows in Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The value of the name attribute of the cfwindow tag for which you want the object.</td>
</tr>
</tbody>
</table>

Returns

An object of type Ext.BasicDialog.

Usage

Use this function to get the Ext JavaScript Library Ext.BasicDialog object that underlies the HTML cfwindow control. You can then use the raw object to modify the displayed window. For documentation on the objects and how to manage them, see the [Ext JavaScript library documentation](http://extjs.com/deploy/ext-1.1.1/docs/).

#back to top
ColdFusion.Window.hide

Description

Hides a window that is currently displayed.

Function syntax

```
ColdFusion.Window.hide(name)
```

See also

cfwindow, ColdFusion.Window.create, ColdFusion.Window.getWindowObject, ColdFusion.Window.onHide, ColdFusion.Window.onShow, ColdFusion.Window.show, ColdFusion.Tree.getTreeObject, Using pop-up windows in Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name attribute of the window to hide.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This tag has no effect if the window is already hidden.

Example

The following code lets you show and hide a window by clicking buttons:
<html xmlns="http://www.w3.org/1999/xhtml">
<head></head>
<body>
<cfwindow name="testWindow" initshow=true title="test window" closable=true>
  Window contents
</cfwindow>
<cfform>
  <cfinput name="hidebutton" type="button" value="Hide Window"
    onclick="javascript:ColdFusion.Window.hide('testWindow');="/>
  <cfinput name="showbutton" type="button" value="Show Window"
    onclick="javascript:ColdFusion.Window.show('testWindow');="/>
</cfform>
</body>
</html>
ColdFusion.Window.onHide

Description

Specifies a function to run each time a specific window hides.

Function syntax

```
ColdFusion.Window.onHide(windowName, handler)
```

See also

cfwindow, ColdFusion.Window.create, ColdFusion.Window.getWindowObject, ColdFusion.Window.hide, ColdFusion.Window.onShow, ColdFusion.Window.show, ColdFusion.Tree.getTreeObject, Using pop-up windows in Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>windowName</td>
<td>The name of the window. The handler function runs whenever this window hides.</td>
</tr>
<tr>
<td>handler</td>
<td>The JavaScript function to run when the window hides.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

The function specified in the `handler` parameter can optionally take one parameter, which contains the window name.

Example

The following example uses the `ColdFusion.Window.onHide` function to display an alert with information about the window when you click a button that hides the window:
<head>
<script language="javascript">
function onHide(name) {
    alert("window hidden = " + name);
}

function test() {
    ColdFusion.Window.onHide("testWindow", onHide);
    ColdFusion.Window.hide("testWindow");
}
</script>
</head>
<body>
<cfwindow name="testWindow" initshow=true title="test window"
closable=true>
    Window contents
</cfwindow>

<cfform>
    <cfinput name="button" value="Hide Window" onclick="javascript:test()"
type="button"/>
</cfform>
</body>
<html>
ColdFusion.Window.onShow

Description

Specifies a function to run each time a specific window shows, including when you create a window and specify an initShow attribute or configuration entry value of true.

Function syntax

ColdFusion.Window.onShow(windowName, handler)

See also

cfwindow, ColdFusion.Window.create, ColdFusion.Window.getWindowObject, ColdFusion.Window.onHide, ColdFusion.Window.hide, ColdFusion.Window.show, ColdFusion.Tree.getTreeObject, Using pop-up windows in Using Ajax form controls and features in Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>windowName</td>
<td>The name of the window. The handler function runs whenever this window shows.</td>
</tr>
<tr>
<td>handler</td>
<td>The JavaScript function to run when the window shows.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

The function specified in the handler parameter can optionally take one parameter, which contains the window name. One use for this function is to fetch window data only when the window shows. You could use a cfajaxproxy tag to create a JavaScript proxy for a CFC function that provides the data, and then a ColdFusion.Window.onShow function to specify a function that calls the proxy function and updates the window contents with the new data.

Example

The following example uses the ColdFusion.Window.onShow function to display an alert with information about the window when you click a button that shows the window:
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script language="javascript">
function onshow(name) {
    alert("window shown = " + name);
}

function test() {
    ColdFusion.Window.onShow("testWindow", onshow);
    ColdFusion.Window.show("testWindow");
}
</script>
</head>
<body>
<cfwindow name="testWindow" initshow=false title="test window"
closable=true>
Window contents
</cfwindow>

cfform
<cfinput name="button" value="show Window" onclick="javascript:test()"
type="button"/>
</cfform>
</body>
</html>
ColdFusion.Window.show

Description

Shows a window that is currently hidden.

Function syntax

```
ColdFusion.Window.show(name)
```

See also

cfwindow, ColdFusion.Window.create, ColdFusion.Window.getWindowObject, ColdFusion.Window.hide, ColdFusion.Window.onHide, ColdFusion.Window.onShow, ColdFusion.Tree.getTreeObject, Using pop-up windows in Using Ajax form controls and features in the Developing ColdFusion Applications

History

ColdFusion 8: Added this function

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name attribute of the window to show.</td>
</tr>
</tbody>
</table>

Returns

This function does not return a value.

Usage

This function shows a window that you created with an initShow attribute or parameter value of false, or that you hid by calling the ColdFusion.Window.hide function. It does not show a window that a user closed by clicking the x icon on the title bar. This function has no effect if the window is already shown.

Example

See the example at ColdFusion.Window.hide

#back to top
JavaScriptFunctionsinColdFusion9Update1

The following are the Ajax JavaScript functions added in this release:

ColdFusion.Autosuggest.getAutosuggestObject

Description

Lets you access underlying YUI AutoComplete object thereby providing fine-grained control over the object, for example attaching an event.

Returns

The underlying AutoComplete object.

Function syntax

ColdFusion.Autosuggest.getAutosuggestObject (Id)

Parameters

- Id: Name of the auto-suggest object.

Example
ColdFusion.Layout.disableSourceBind

Description

Disables the source bind.

Function syntax

ColdFusion.Layout.disableSourceBind(Id)

Parameters

- Id: Name of the layout area.
**Usage**

Assume that you are using `Coldfusion.navigate` to populate content into tab or accordion panels. You can have instances where content comes from the source bind call if the `source` attribute is defined for `cflayoutarea` (and is not from `ColdFusion.navigate`). In such instances, you might disable the source bind to get content using `Coldfusion.navigate`.

**Example**

layout.cfm uses the templates Tab1_Src.cfm, Tab2_Src.cfm, and Tab3_Src.cfm. If you run layout.cfm, you notice that clicking

- `navigate` populates content of `tab2_src.cfm` instead of `navigate.cfm`
- Disable Source bind ensures that the content of `navigate.cfm` is populated in `tab2_src`
- Enable Source Bind and then clicking `tab2_src` would again populate the content of `tab2_src`

**Tab1_Src.cfm**

```
<br><cfdump var="#CGI#" keys="15" label="[CGI scope]"><br>
```

**Tab2_Src.cfm**

```
<br><cfdump var="#server#" label="[Server scope]"><br>
```

**Tab3_Src.cfm**

```
<br><cfdump var="#server.coldfusion#" label="[Showing key coldfusion in server scope]"><br>
```

**Tab4_Src.cfm**

```
<br><cfdump var="#server.os#" label="[Showing key OS in server scope]"><br>
```

**layout.cfm**


ColdFusion.Layout.enableSourceBind

*Description*

If disabled, enables the source bind.
Function syntax

ColdFusion.Layout.enableSourceBind( Id )

Parameters

- Id: Name of the layout area.

Usage

See usage in ColdFusion.Layout.disableSourceBind.

Example

See example in ColdFusion.Layout.disableSourceBind.

ColdFusion.FileUpload.getSelectedFiles

Description

Returns an array of objects containing the filename and size of the files selected for upload. The file size is returned in bytes. The function also returns file upload status as YES|NO|Error.

Function syntax

ColdFusion.FileUpload.getSelectedFiles( Id )

Parameters

- Id: Name of the cffileupload control.

Usage

In a real life scenario, you normally use the uploader with other controls. For example, a form with three fields: name, email, and uploader. Assume that you upload the files, but forget to click Submit or you select the files, submit the form, but forget to click Upload. You can use this function to inform the user that there are files that have been selected for upload and provide the following details:

- FILENAME: Name of the file selected for upload.
- SIZE: Size of the file in bytes.
- STATUS: YES|NO|Error; YES indicates a successful upload, NO indicates that the upload is yet to occur, and Error indicates that an exception has occurred during the upload operation.

Example

The following example illustrates a scenario where the user clicks Submit and is informed about the files selected for upload:
Coldfusion.fileUpload.setUrl

Description

Used to set URL for the fileupload control dynamically.

Returns

Nothing

Function syntax

<cffileupload
    url="uploadAll.cfm"
    name="myuploader1"
    hideUploadButton=false
    onUploadComplete="foo"
/>
ColdFusion.fileUpload.setUrl(id, url)

**Parameters**

- **Id**: Name of upload control.
- **Url**: URL can be an absolute URL, relative URL, or fully qualified URL.

**Example**

```javascript
<script language="javascript">
  var uploadDone = function(result){
    alert("File uploaded");
  }

  var setUploadUrl = function(id)
  {
    var selectedFiles = ColdFusion.FileUpload.getSelectedFiles(id);
    var uploadUrl = "/manual/ajaxui/cffileupload/setUrl/includes/_uploadall.cfm";
    alert("Upload URL : " + uploadUrl);
    if(selectedFiles.length){
      ColdFusion.FileUpload.setURL(id,uploadUrl);
      ColdFusion.FileUpload.startUpload(id);
    }
  }

  var callbackhandler = function(obj)
  {
    var fileName = obj["FILENAME"];
    var status = obj["STATUS"];
    var message = obj["MESSAGE"];
    var msg = "In callbackhandler()" + "\n\n" + "FILENAME: " + fileName + "\n\n" + "STATUS: " + status + "\n\n" + "MESSAGE: " + message
    alert(msg);
  }

  var errorhandler = function()
  {
    alert("In errorhandler()");
  }

  var uploadcompleted = function()
  {
    alert("All files have been uploaded successfully");
  }
</script>

<cfform name="frmUpload">
  <br>
  <cffileupload name="uploader" hideuploadbutton="true" onComplete="uploadDone" onError="errorhandler" onUploadComplete="uploadcompleted">
    <br>
    <cfinput type="button" name="submit" value="Click to set URL and Upload Files" onClick="setUploadUrl('uploader')">
  </cfform>
```

ColdFusion.grid.getSelectedRows
**Description**

Used to fetch data for the selected rows in the grid.

**Returns**

An array of objects that contains row data.

**Function syntax**

```
ColdFusion.grid.getSelectedRows(id)
```

**Parameters**

- `id`: Name of the grid defined using `cfgrid`.

**See also**

FileUpload

**Usage**

See the example in `ColdFusion.grid.clearSelectedRows`.

**Example**

See the example in `ColdFusion.grid.clearSelectedRows`.

**ColdFusion.grid.clearSelectedRows**

**Description**

Used to clear the selected rows in the grid.

**Returns**

Nothing

**Function syntax**

```
ColdFusion.grid.clearSelectedRows(id)
```

**Parameters**

- `id`: Name of the grid defined using `cfgrid`.

**Usage**

See the following example.

**Example**

Employee.cfm

```html
<html>
<head>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```
<cfajaxproxy cfc="emp" jclassname="emputils">
<cfscript language="javascript">
var emp = new emputils();
var deleteAllSelectedRows = function(grid)
{
    emp.setHTTPMethod("POST");
    emp.deleteEmployees(getAllSelectedRows(grid,false));
    ColdFusion.Grid.refresh(grid);
}
var getAllSelectedRows = function(grid,showalert)
{
    obj = ColdFusion.Grid.getSelectedRows(grid);
    jsonbj = ColdFusion.JSON.encode(obj);
    if(showalert)
        alert(jsonbj);
    return obj;
}
var clearAllSelectedRows = function(grid)
{
    ColdFusion.Grid.clearSelectedRows(grid);
}
</script>
</head>
<body>
<cfform>
<cfgrid format="html"
name="empListing"
selectmode="edit"
bind="cfc:emp.getEmployees({cfgridpage},{cfgridpagesize},{cfgridsortcolumn},{cfgridsortdirection})"
 onchange="cfc:emp.editEmployees({cfgridaction},{cfgridrow},{cfgridchanged})"
autowidth="true"
multirowselect=true
delete="true"
insert="true"
title="Employee database"
pagesize="25"
>
<cfgridcolumn name="EMP_ID" header="EMP_ID" select="false" display="false">
<cfgridcolumn name="FIRSTNAME" header="First Name" select="true" />
<cfgridcolumn name="LASTNAME" header="Last Name" select="true" />
<cfgridcolumn name="DEPARTMENT" header="Department" select="true" />
<cfgridcolumn name="EMAIL" header="Email" select="true" />
</cfgrid>
<br>
<cfinput type="button" onClick="javascript:getAllSelectedRows('empListing',true)"
name="getRows" value="Get Selected Rows">
<cfinput type="button" onClick="javascript:clearAllSelectedRows('empListing')"
name="clearRows" value="Clear Selected Rows">
<cfinput type="button" onClick="javascript:deleteAllSelectedRows('empListing')"
name="deleteRows" value="Delete Selected Rows">
</cfform>
<cfcomponent>
    <cfscript>
        empQuery = new query(name="emps", datasource="cfdocexamples");
        remote any function
        getEmployees(page,pageSize,gridsortcolumn="EMP_ID",gridsortdirection="ASC",empName)
        {
            var orderBy = "EMP_ID";
            var mysql = "SELECT Emp_ID, FirstName, LastName, EMail, Department, Email FROM Employees"
            if(isdefined("arguments.empName") and trim(arguments.empName) neq "){
                mysql = mysql & " WHERE " & "firstname = '#arguments.empName#'";
            }
            if(arguments.gridsortcolumn eq "){
                mysql = mysql & " ORDER BY " & orderBy;
            }
            mysql = mysql & " " & gridsortdirection;
            return QueryConvertForGrid(empQuery.execute(sql=mysql).getResult(), page, pageSize);
        }
        remote void function editEmployees(gridaction,gridrow,gridchanged)
        {
            switch(gridaction)
            {
            case "I":
                {
                    var eFName = gridrow["FIRSTNAME"]; 
                    var eLName = gridrow["LASTNAME"]; 
                    var eDept = gridrow["DEPARTMENT"]; 
                    var eEmail = gridrow["EMAIL"]; 
                    var insertSql = "insert into Employees(FirstName,LastName,Department,Email) values (" & "/" & eFName & "/", "/" & eLName & "/", "/" & eDept & "/", "/" & eEmail & "/")";
                    empQuery.execute(sql=insertSql);
                    break;
                }
            case "U":
                {
                    var empId = gridrow["EMP_ID"]; 
                    var changedCol = structkeylist(gridchanged);
                    var updateSql = "UPDATE Employees SET " & changedCol & "/" & gridchanged[changedCol] & "/" WHERE emp_id=" & empId;
                    empQuery.execute(sql=updateSql);
                    break;
                }
            case "D":
                {
                    deleteEmployees(gridrow);
                }
            }
        }
        remote void function deleteEmployees(empdata)
    </cfscript>
</cfcomponent>
{ 
    var i = 1;
    var emp = {};
    if(isArray(empdata) and not ArrayIsEmpty(empdata)){
        for(emp in empdata){
            if(isStruct(emp) and structkeyexists(emp,"emp_id"){
                empid = emp["emp_id"]; 
                writeln("deleting " & empid);
                //var deleteSql = "delete from Employees where emp_id=" & empid;
                //empQuery.execute(sql=deleteSql);
            }
        }
    }
}
In this example, setting `multirowselect=true` enables performing of batch operations on grid data, such as deleting multiple records. In the `deleteemployees` functions, two lines have been commented out to prevent accidental deletion of data (since it is a batch operation). To see deletion, uncomment the code. The form has a `deleteAllSelectedRows` button that illustrates how records can be deleted externally. That is, without using the `delete` button built in to the grid. The same approach can be used to perform other batch operations such as moving multiple files to another folder or batch updates.

⚠️ **Note**

Set the `httpMethod` to POST on the Proxy object carefully to avoid "request URI too large" errors as shown in the `deleteAllSelectedRows` method in `Employee.cfm`.

---

**ColdFusion.Map.show**

*Description*

Shows the map if it is hidden.

*Function syntax*

`ColdFusion.Map.show(Id)`

*Parameters*

- `Id`: Name of the map.

*Example*
ColdFusion.Map.hide

Description

If displayed, hides the map.

Function syntax

ColdFusion.Map.hide(Id)

Parameters

- Id: Name of the map.

Example

See example in ColdFusion.Map.show

ColdFusion.Map.refresh

Description

Reloads the map.

Function syntax

ColdFusion.Map.refresh (Id)

Parameters

- Id: Name of the map.
Usage

If the map is embedded within spry collapsible panels or divs that are hidden on display, that is the map container is displayed while the actual map is hidden, use this function to force the map to display.

Example

```html
<cfmap
width="500"
height="200"
zoomlevel="12"
name="mainMap"
markercolor="333444"
showscale="false"
typecontrol="none"
showcentermarker="true"
centeraddress="The Key Learning centre, Oxford, UK"
>
</cfmap>
</div>
</script>

ColdFusion.Grid.getTopToolbar

Description

Gets the top toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.getTopToolbar(Id)

Parameters

- **Id**: Name of the grid.

Example
ColdFusion.Grid.getBottomToolbar

*Description*

Gets bottom toolbar that can be used to add a control, for example icon or button.

*Function syntax*

ColdFusion.Grid.getBottomToolbar(Id)

*Parameters*

- Id: Name of the grid.

*Example*

See example in ColdFusion.Grid.refreshBottomToolbar.

ColdFusion.Grid.showTopToolbar

*Description*

Displays the top toolbar that can be used to add a control, for example icon or button.

*Function syntax*

ColdFusion.Grid.showTopToolbar(Id)

*Parameters*

- Id: Name of the grid.

*Example*

See example in ColdFusion.Grid.refreshBottomToolbar.

ColdFusion.Grid.hideTopToolbar

*Description*

Hides the top toolbar that can be used to add a control, for example icon or button.

*Function syntax*

ColdFusion.Grid.hideTopToolbar(Id)

*Parameters*

- Id: Name of the grid.

*Example*

See example in ColdFusion.Grid.refreshBottomToolbar.

ColdFusion.Grid.showBottomToolbar
Description

Shows bottom toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.showBottomToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar.

ColdFusion.Grid.hideBottomToolbar

Description

Hides the bottom toolbar that can be used to add a control, for example icon or button.

Function syntax

ColdFusion.Grid.hideBottomToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar.

ColdFusion.Grid.refreshTopToolbar

Description

Refreshes the top toolbar that can be used to add a control, for example icon or button. This function internally calls the JavaScript function ColdFusion.Grid.showTopToolbar.

Function syntax

ColdFusion.Grid.refreshTopToolbar(Id)

Parameters

- Id: Name of the grid.

Example

See example in ColdFusion.Grid.refreshBottomToolbar.
calls the JavaScript function ColdFusion.Grid.showBottomToolbar.

**Function syntax**

ColdFusion.Grid.refreshBottomToolbar(\text{Id})

**Parameters**

- \text{Id}: Name of the grid control.

**Example**

grid.cfc

```cfc
<cfcomponent>
<cfscript>
remote any function
getEmployees(page, pageSize, gridsortcolumn="EMP_ID", gridsortdirection="ASC"){
    var startRow = (page-1)*pageSize;
    var endRow = page*pageSize;

    if(!isdefined("arguments.gridsortcolumn") or
        isdefined("arguments.gridsortcolumn") and trim(arguments.gridsortcolumn) eq "")
        gridsortcolumn = "EMP_ID";
    if(!isdefined("arguments.gridsortdirection") or
        isdefined("arguments.gridsortdirection") and arguments.gridsortdirection eq "")
        gridsortdirection = "ASC";
    var mysql = "SELECT Emp_ID, FirstName, EMail, Department FROM Employees";
    if(isdefined("arguments.gridsortcolumn") and
        arguments.gridsortcolumn neq "")
        mysql = mysql & " ORDER By " & gridsortcolumn;
    if(isdefined("arguments.gridsortdirection") and
        arguments.gridsortdirection neq "")
        mysql = mysql & " " & gridsortdirection ;
    rs1 = new query(name="team", datasource="cfdocexamples", sql=mysql).execute();
    return QueryConvertForGrid(rs1.getResult(), page, pageSize);
}

remote any function editEmployees(gridaction, gridrow, gridchanged){
    writelog("edit employee info");
}
</cfscript>
</cfcomponent>
```

grid.cfm

```cfc
<script>
    var refreshToolbar = function(id, type){
        type == "top" ? ColdFusion.Grid.refreshTopToolbar(id) :
                     ColdFusion.Grid.refreshBottomToolbar(id);
```
```javascript
var hideToolbar = function(id,type){
    type == "top" ? ColdFusion.Grid.hideTopToolbar(id) : ColdFusion.Grid.hideBottomToolbar(id);
}

var showToolbar = function(id,type){
    (type == "top") ? ColdFusion.Grid.showTopToolbar(id) : ColdFusion.Grid.showBottomToolbar(id);
}

var handleToolbar = function(id,type){
    if(type == "top"){
        tbar = ColdFusion.Grid.getTopToolbar(id);
        tbar.addButton({
            text: "Add User Account",
            tooltip: "Add a user account",
            handler: addUserAccount
        });
    }
    else{
        bbar = ColdFusion.Grid.getBottomToolbar(id);
        bbar.add(new Ext.Toolbar.Separator());
        bbar.addButton({
            text: "Delete User Account",
            tooltip: "Delete a user account",
            handler: deleteUserAccount
        });
    }
}

var GetUserInfo = function(){
    alert("Retrieving user account");
}

var addUserAccount = function(){
    alert("Adding new user account")
}
var deleteUserAccount = function(){
    alert("Deleting user account")
}
</script>
</cfform>
<br>
</cfinput type="button" onClick="showToolbar('empGrid','top')" name="btn1" value="Show Top Toolbar">
<cfinput type="button" onClick="handleToolbar('empGrid','top')" name="btn2" value="Add button to Top Toolbar">
<cfinput type="button" onClick="refreshToolbar('empGrid','top')" name="btn3" value="Refresh Top Toolbar">
<cfinput type="button" onClick="hideToolbar('empGrid','top')" name="btn4" value="Hide Top Toolbar">
<br><br>
<cfgrid format="html"
    name="empGrid"
    width="800"
```
pagesize=5
sort=true
title="Employee database"
collapsible="true"
insert="yes"
delete="yes"

bind="cfc:grid.getEmployees({cfgridpage},{cfgridpagesize},{cfgridsortcolumn},{cfgridsortdirection})"

onChange="cfc:grid.editEmployees({cfgridaction},{cfgridrow},{cfgridchanged})"
selectMode="edit"

<cfgridcolumn name="Emp_ID" display=false header="ID" />
<cfgridcolumn name="FirstName" display=true header="First Name" />
<cfgridcolumn name="Email" display=true header="Email" />
<cfgridcolumn name="Department" display=true header="Department" />

</cfgrid>

<br><br>
<cfinput type="button" onClick="hideToolbar('empGrid','bottom')" name="btn5" value="Hide Bottom Toolbar">
<cfinput type="button" onClick="showToolbar('empGrid','bottom')" name="btn6" value="Show Bottom Toolbar">
<cfinput type="button" onClick="handleToolbar('empGrid','bottom')" name="btn7" value="Add button to Bottom Toolbar">
<cfinput type="button" onClick="refreshToolbar('empGrid','bottom')"
• **ColdFusion.Autosuggest.getAutosuggestObject**
  - Description
  - Returns
  - Function syntax
  - Parameters
  - Example

• **ColdFusion.Layout.disableSourceBind**
  - Description
  - Function syntax
  - Parameters
  - Usage
  - Example

• **ColdFusion.Layout.enableSourceBind**
  - Description
  - Function syntax
  - Parameters
  - Usage
  - Example

• **Coldfusion.fileUpload.setUrl**
  - Description
  - Returns
  - Function syntax
  - Parameters
  - Usage
  - Example

• **ColdFusion.grid.getSelectedRows**
  - Description
  - Returns
  - Function syntax
  - Parameters
  - See also
  - Usage
  - Example

• **ColdFusion.grid.clearSelectedRows**
  - Description
  - Returns
  - Function syntax
  - Parameters
  - Usage
  - Example
- **ColdFusion.Map.show**
  - **Description**
  - **Function syntax**
  - **Parameters**
  - **Example**

- **ColdFusion.Map.hide**
  - **Description**
  - **Function syntax**
  - **Parameters**
  - **Example**

- **ColdFusion.Map.refresh**
  - **Description**
  - **Function syntax**
  - **Parameters**
  - **Usage**
  - **Example**

- **ColdFusion.Grid.getTopToolbar**
  - **Description**
  - **Function syntax**
  - **Parameters**
  - **Example**

- **ColdFusion.Grid.getBottomToolbar**
  - **Description**
  - **Function syntax**
  - **Parameters**
  - **Example**

- **ColdFusion.Grid.showTopToolbar**
  - **Description**
  - **Function syntax**
  - **Parameters**
  - **Example**

- **ColdFusion.Grid.hideTopToolbar**
  - **Description**
  - **Function syntax**
  - **Parameters**
  - **Example**

- **ColdFusion.Grid.showBottomToolbar**
  - **Description**
  - **Function syntax**
  - **Parameters**
  - **Example**

- **ColdFusion.Grid.hideBottomToolbar**
  - **Description**
  - **Function syntax**
  - **Parameters**
  - **Example**

- **ColdFusion.Grid.refreshTopToolbar**
Script Functions Implemented as CFCs

Script functions were added in ColdFusion 9. They are implemented as ColdFusion Components. These functions extend the usage of the tags `cfmail, cfpdf, cfquery, cfhttp, cfstoredproc, and cfftp` to CFScript.

- Accessing the functions
- Function summary
- ftp
- http
- mail
- pdf
- query
- storedproc
- Script functions implemented as CFCs in ColdFusion 9 Update 1

#back to top
Accessing the functions

Script functions are available in the following location: cf_root\CustomTags\com\adobe\coldfusion. Ensure that you do not delete the default custom tag mapping listed in the ColdFusion Administrator (Extensions > Custom Tag Paths > Custom tag mappings).

Script functions work if they are either in the default location or web root. If you have the functions in any other location, add a /com mapping in the ColdFusion Administrator that points to the new location (for example C:\com).

⚠️ Note

Values of the attributes set in a service action, for example, `mail.send(body="test mail")` are transient in nature. They are not accessible after the action completes. Accessing the attributes using implicit getters results in error whereas any attributes set using either implicit setters or the init method call are retained and can be accessed using implicit getters.
**Function summary**

The following table lists the script functions and the equivalent ColdFusion tag.

<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent ColdFusion Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp</td>
<td>cfftp</td>
</tr>
<tr>
<td>http</td>
<td>cfhttp</td>
</tr>
<tr>
<td>mail</td>
<td>cfmail</td>
</tr>
<tr>
<td>pdf</td>
<td>cfpdf</td>
</tr>
<tr>
<td>query</td>
<td>cfquery</td>
</tr>
<tr>
<td>storedproc</td>
<td>cfstoredproc</td>
</tr>
</tbody>
</table>

#back to top
**ftp**

**Description**

Used to implement File Transfer Protocol (FTP) operations using CFScript.

**Syntax**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td><code>new ftp()</code> or <code>createObject(&quot;component&quot;,&quot;ftp&quot;)</code></td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>Any one of the following:</td>
</tr>
<tr>
<td></td>
<td>• <code>ftpService=new ftp(attribute-value pair)</code></td>
</tr>
<tr>
<td></td>
<td>• <code>ftpService.setAttributes(_attribute-value pair_)</code></td>
</tr>
<tr>
<td></td>
<td>• <code>ftpService.setAttributeName_(attribute_value)</code></td>
</tr>
<tr>
<td></td>
<td>• <code>ftpService.action_method(attribute-value_pair)</code></td>
</tr>
<tr>
<td>Executing the service action</td>
<td><code>ftpService.action_method(attribute-value_pair)</code></td>
</tr>
</tbody>
</table>

**Properties**

<table>
<thead>
<tr>
<th>actionparam</th>
<th>buffersize</th>
<th>connection</th>
<th>passive</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>port</td>
<td>proxyserver</td>
<td>retrycount</td>
</tr>
<tr>
<td>server</td>
<td>stoponerror</td>
<td>timeout</td>
<td>username</td>
</tr>
<tr>
<td>fingerprint</td>
<td>key</td>
<td>passphrase</td>
<td>secure</td>
</tr>
<tr>
<td>ASCIIExtensionList</td>
<td>directory</td>
<td>existing</td>
<td>failifexists</td>
</tr>
<tr>
<td>item</td>
<td>localfile</td>
<td>name</td>
<td>new</td>
</tr>
<tr>
<td>remotefile</td>
<td>result</td>
<td>transfermode</td>
<td>allosize</td>
</tr>
</tbody>
</table>

All attributes supported by the tag `cfftp` can be used as attribute-value pairs. For example,

```
<cfftp userName="myUserName"/>
```
can be used as

```javascript
ftpService.setUserName("myUserName");
```

For details, see the Attributes section for the `cfftp` tag.

**See also**

`cfftp`, Function summary

**History**

ColdFusion 9: Added this function.

**Methods**

The following FTP actions are available as methods. All methods have similar arguments and syntax.

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>close</td>
</tr>
<tr>
<td>allo</td>
<td>acct</td>
</tr>
<tr>
<td>listDir</td>
<td>removeDir</td>
</tr>
<tr>
<td>rename</td>
<td>remove</td>
</tr>
<tr>
<td>existDir</td>
<td>existsFile</td>
</tr>
</tbody>
</table>

**Description**

All methods correspond to the FTP actions supported by the tag `cfftp`. For details of each method, refer to the relevant section for the tag `cfftp`.

**Returns**

All methods return a component with the following properties set:

- **prefix**: Equivalent to the `result` attribute or `cfftp scope`
- **result**: Applicable only for `action=listdir`

**Syntax**

```
ftpService.methodName(attribute-value pair)
```

**Arguments**

All attributes supported by the tag `cfftp`.

**setAttributes**

**Description**

Sets attributes for the `ftp` function.
### Returns
Nothing

### Syntax
ftpService.setAttributes (attribute-value pair)

### Arguments
All attributes supported by the tag `cfftp`.

### getAttributes

<table>
<thead>
<tr>
<th>Description</th>
<th>Gets the attributes that were set for the <code>ftp</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Returns a struct with all or some attribute values.</td>
</tr>
<tr>
<td>Syntax</td>
<td>ftpService.get_Attributes_ (attribute list)</td>
</tr>
<tr>
<td>Arguments</td>
<td>A comma-separated list of attributes. If no list is specified, all defined attributes are returned.</td>
</tr>
</tbody>
</table>

### clear

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes all attributes added for the <code>ftp</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td>ftpService.clear()</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
</tbody>
</table>

### Usage
This function corresponds to the `cfftp` tag. For details, see the Usage section for the tag `cfftp`.

### Example
<cfscript>
    /* Create a new ftp Service*/
    ftpService = new ftp();
    /* Set attributes using implicit setters */
    ftpService.setUsername("myUsername");
    ftpService.setPassword("myPassword");
    ftpService.setServer("myFtpServer");
    ftpService.setStopOnError("true");
    ftpService.setConnection("conn");
    /* Open connection to ftp server */
    WritesOutput("<h4>Open a connection</h4>");
    result = ftpService.open();
    WriteOutput("<p>Did it succeed? " & result.getPrefix().succeeded & "<br>")
    /* Get current directory */
    WriteOutput("<h4>Get current directory</h4>");
    result = ftpService.getCurrentDIR();
    WriteOutput("<p>Current Directory: " & result.getPrefix().returnvalue & "<br>");
    /* List contents of the current directory */
    WriteOutput("<h4>List directory contents</h4>");
    result = ftpService.listdir(directory = "/", name="listDirs");
    displayListing(result.getResult());
    /* Move a file to the ftp server */
    WriteOutput("<h4>Move File to Remote Server</h4>");
    lFile = "C:\temp\artifacts.xml";
    rFile = "artifacts.xml";
    result = ftpService.putFile(transferMode="binary", localfile=lFile,
    remoteFile=rFile);
    WriteOutput("<p>Did it succeed? " & result.getPrefix().succeeded & "<br>");
    /* Close connection to the ftp server */
    WriteOutput("<h4>Close the connection</h4>");
    ftpService.close(connection="conn");
    WriteOutput("<p>Did it succeed? " & result.getPrefix().succeeded & "<br>");
    </cfscript>
<cffunction name="displayListing" hint="display ftp files">
    <cfargument name="filesToList" required="true">
    <cftable query = "filesToList" HTMLTable = "Yes" colHeaders = "Yes" border="1"
    maxrows="10">
        <cfcol header = "<b>Name</b>" text = "#name#">
        <cfcol header = "<b>Path</b>" text = "#path#">
        <cfcol header = "<b>URL</b>" text = "#url#">
        <cfcol header = "<b>Length</b>" text = "#length#">
        <cfcol header = "<b>Last Modified</b>" text = "#DateFormat(lastmodified)#">
        <cfcol header = "<b>Is Directory</b>" text = "#isdirectory#">
    </cftable>
</cffunction>
http

Description

Used in CFScript to generate an HTTP request and handle the response from the server.

Syntax

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td>new http() or createObject(&quot;component&quot;,&quot;http&quot;)</td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>Any one of the following:</td>
</tr>
<tr>
<td></td>
<td>* httpService=new http(<em>attribute-value_pair</em>)</td>
</tr>
<tr>
<td></td>
<td>* httpService.setAttributes(<em>attribute-value_pair</em>)</td>
</tr>
<tr>
<td></td>
<td>* httpService.set_AttributeName_(attribute_value)</td>
</tr>
<tr>
<td></td>
<td>* httpService.send(attribute-value_pair)</td>
</tr>
<tr>
<td>Executing the service action</td>
<td>httpService.send(<em>attribute-value_pair</em>)</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>url</th>
<th>charset</th>
<th>clientcert</th>
<th>clientcertpassword</th>
</tr>
</thead>
<tbody>
<tr>
<td>columns</td>
<td>delimiter</td>
<td>file</td>
<td>firstrowasheaders</td>
</tr>
<tr>
<td>getasbinary</td>
<td>method</td>
<td>multipart</td>
<td>multiparttype</td>
</tr>
<tr>
<td>name</td>
<td>password</td>
<td>path</td>
<td>port</td>
</tr>
<tr>
<td>proxyserver</td>
<td>proxypart</td>
<td>proxyuser</td>
<td>proxypassword</td>
</tr>
<tr>
<td>redirect</td>
<td>resolveurl</td>
<td>result</td>
<td>textqualifier</td>
</tr>
<tr>
<td>throwonerror</td>
<td>timeout</td>
<td>useragent</td>
<td>username</td>
</tr>
</tbody>
</table>

All attributes supported by the tag <cfhttp> can be used as attribute-value pairs. For example,

```
<cfhttp name="onerow">
```

can be used as
httpService.setName("onerow");

For details of the attributes, see the Attributes section for the tag `cfhttp`.

See also

`cfhttp`, Function summary

History

ColdFusion 9: Added this function.

Methods

- `addParam`

<table>
<thead>
<tr>
<th>Description</th>
<th>Used to add <code>cfhtpparam</code> tags. For example, to specify http POST operations in CFScript. Specifies parameters to build an HTTP request.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>httpService.addParam(attribute-value)</code></td>
</tr>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by <code>cfhtpparam</code> tag can be used as attribute-value pairs.</td>
</tr>
</tbody>
</table>

- `send`

<table>
<thead>
<tr>
<th>Description</th>
<th>Used to generate an HTTP request and handle the response from the server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>A component on which the following methods can be invoked:</td>
</tr>
<tr>
<td></td>
<td>- <code>GetResult()</code>: To access the query object returned by the server if a name attribute is specified.</td>
</tr>
<tr>
<td></td>
<td>- <code>GetPrefix()</code>: To access the <code>cfhttp</code> scope. This is equivalent to the <code>result</code> attribute of the <code>cfhttp</code> tag.</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>httpService.send(attribute_value_pair)</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by the <code>cfhtpparam</code> tag.</td>
</tr>
</tbody>
</table>

- `setAttributes`

<table>
<thead>
<tr>
<th>Description</th>
<th>Sets attributes for the <code>http</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getAttributes</td>
<td>Gets attributes that were set for the http function.</td>
</tr>
<tr>
<td>clearAttributes</td>
<td>Removes all attributes added for the http function.</td>
</tr>
<tr>
<td>clearParams</td>
<td>Removes cfhttpparam} tags that were added using the {{addParam method.</td>
</tr>
<tr>
<td>clear</td>
<td>Removes all attributes and cfhttpparam} tags that were added using the addParam method.</td>
</tr>
</tbody>
</table>
### Returns
Nothing

### Syntax
`httpService.clear()`

### Arguments
None

## Usage
This function corresponds to the `cfhttp` tag. For usage details, see the Usage section for `cfhttp` in the CFML Reference.

## Example
```cfscript
<!--- Get Video --->
<cfset videoName = "<video path>\hello.wmv">  
<cfset videoFileName = "hello.wmv">  
--->
<!--- Set User Account Data --->
<cfset clientKey = "enter client key from google"/>  
<cfset devKey = "enter dev key from google"/>  
--->
<cfscript>
/* youtube upload url */
youtubeUploadURL = "http://uploads.gdata.youtube.com/feeds/api/users/default/uploads";
/* video to upload */
videoName = ExpandPath('./hello.wmv');
videoFileName = "hello.wmv";
/* set user account data */
clientKey = "enter client key from google";
devKey = "enter dev key from google";
/* create new http service */
httpService = new http();
/* set attributes using implicit setters */
httpService.setMethod("post");
httpService.setCharset("utf-8");
httpService.setUrl("https://www.google.com/accounts/ClientLogin");
/* add httpparams using addParam() */
httpService.addParam(type="formfield",name="accountType",value="HOSTED_OR_GOOGLE");
httpService.addParam(type="formfield",name="Email",value="enter gmail id");
httpService.addParam(type="formfield",name="Passwd",value="enter password");
httpService.addParam(type="formfield",name="service",value="youtube");
httpService.addParam(type="formfield",name="source",value="youtubecode");
/* make the http call to the URL using send() */
result = httpService.send().getPrefix();
/* process the filecontent returned */
content = listtoarray(result.filecontent,chr(10));
for(i=1;i <= arraylen(content);i++)
{
  item = content[i];
  authdata[listFirst(item, "=")] = listRest(item, "=");
```
<cfscript>
	tmpfile = expandPath("./meta.xml");
	FileWrite(tmpfile, trim(meta));
	/* use the httpService created above */
	httpService.setUrl("http://uploads.gdata.youtube.com/feeds/api/users/default/uploads");
	httpService.setTimeout(450);
	httpService.setMultipartType("related");
	/* clear params first */
	httpService.clearParams();
	/* add httpparams using addParam() */
	httpService.addParam(type="header", name="Authorization", value="GoogleLogin auth=#authdata.auth#");
	httpService.addParam(type="header", name="X-GData-Client", value="#variables.clientkey#");
	httpService.addParam(type="header", name="X-GData-Key", value="key=#variables.devkey#");
	httpService.addParam(type="header", name="Slug", value="#videoFileName#");
	httpService.addParam(type="file", name="API_XML_Request", file="#tmpfile#", mimetype="application/atom+xml");
	httpService.addParam(type="file", name="file", file="#videoName#", mimetype="video/*");
	/* make the http call to the URL using send() */
	result = httpService.send().getPrefix();
	if(result.statusCode contains "201")
	{
	WriteOutput("Your video has been successfully uploaded to YouTube");
	}
	else
	{
	WriteOutput("There was a problem uploading the video. Status code returned was " & result.statusCode);
	}</cfscript>
# mail

## Description

Used to sends an e-mail message, that optionally contains query output, using an SMTP server.

## Syntax

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td><code>new mail()</code> or <code>createObject(&quot;component&quot;, &quot;mail&quot;)</code></td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>Any one of the following:</td>
</tr>
<tr>
<td></td>
<td>- <code>mailService=new mail(_attribute-value_pair_)</code></td>
</tr>
<tr>
<td></td>
<td>- <code>mailService.setAttributes(_attribute-value_pair_)</code></td>
</tr>
<tr>
<td></td>
<td>- <code>mailService.setAttributeName_(attribute_value)</code></td>
</tr>
<tr>
<td></td>
<td>- <code>mailService.send(attribute-value_pair)</code></td>
</tr>
<tr>
<td>Executing the service action</td>
<td><code>mailService.send(_attribute-value_pair_)</code></td>
</tr>
</tbody>
</table>

## Properties

<table>
<thead>
<tr>
<th>from</th>
<th>to</th>
<th>subject</th>
<th>bcc</th>
<th>cc</th>
<th>charset</th>
<th>debug</th>
<th>failto</th>
<th>group</th>
<th>groupcasesensitive</th>
<th>mailerid</th>
<th>maxrows</th>
<th>mimeattach</th>
<th>password</th>
<th>port</th>
<th>priority</th>
<th>query</th>
<th>replyto</th>
<th>server</th>
<th>spoolenable</th>
<th>startrow</th>
<th>timeout</th>
<th>type</th>
<th>username</th>
<th>useSSL</th>
<th>useTLS</th>
<th>wraptext</th>
<th>remove</th>
<th>body</th>
</tr>
</thead>
</table>

All attributes supported by the tag `cfmail` can be used as attribute-value pairs. For example,

```cfml
<cfmail from="#form.mailFrom#">
```
can be used as

```
mailerService.setFrom(form.mailFrom);
```

See also

cfmail, Function summary

History

ColdFusion 9: Added this function.

Methods

- **addParam**

<table>
<thead>
<tr>
<th>Description</th>
<th>Used to add <code>cfmailparam</code> tags. For example, to attach a file or add a header to an e-mail message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>mailService.addParam(attribute-value pair)</code></td>
</tr>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by the <code>cfmailparam</code> tag can be used as attribute-value pairs.</td>
</tr>
</tbody>
</table>

- **addPart**

<table>
<thead>
<tr>
<th>Description</th>
<th>Used to add <code>cfmailpart</code> tags. For example, one part of a multipart e-mail message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>mailService.addPart(attribute-value pair)</code></td>
</tr>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by the <code>cfmailpart</code> tag can be used as attribute-value pairs.</td>
</tr>
</tbody>
</table>

- **send**

<table>
<thead>
<tr>
<th>Description</th>
<th>Used to invoke the mail service to send an e-mail message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>mailService.send(attribute-value pair)</code></td>
</tr>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
## Arguments

| Arguments | All attributes supported by the `cfmail` tag. |

## setAttributes

| Description | Sets attributes for the `mail` function. |
| Returns | Nothing |
| Syntax | `mailService.setAttributes (attribute-value pair)` |
| Arguments | All attributes supported by the `cfmail` tag. |

## getAttributes

| Description | Gets attributes that were set for the `mail` function. |
| Returns | Returns a struct with all or some of the attribute values. |
| Syntax | `mailService.get_Attributes_ (attribute-list)` |
| Arguments | A comma-separated list of attributes. If no list is specified, all defined attributes are returned. |

## clearAttributes

| Description | Removes all attributes added for the `mail` function. |
| Returns | Nothing |
| Syntax | `mailService.clearAttributes(attribute-list)` |
| Arguments | A comma-separated list of attributes. |

## clearParams

| Description | Removes `cfmailparam` tags that were added using the `addParam` method. |
| Returns | Nothing |
| Syntax | `mailService.clearParams()` |
| Arguments | None |
**clearParts**

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes <code>cfmailpart</code> tags that were added using the <code>addPart</code> method.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>mailService.clearProcResults()</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
</tbody>
</table>

**clear**

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes all attributes, <code>cfmailparam</code> tags, and <code>cfmailpart</code> tags that were added using the methods <code>addParam</code> and <code>addPart</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>mailService.clear()</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
</tbody>
</table>

### Usage

This function corresponds to the tag `cfmail`. For usage details, see the Usage section for `cfmail`.

### Example

```cfscript
<h3>Sending mail in cfscript</h3>
<cfscript>
  /* create mailer service */
  mailerService = new mail();
  if(IsDefined("form.mailto"))
  {
    if(form.mailto is not "" AND form.mailfrom is not "" AND form.Subject is not ""
    and form.attachment is not ")
    {
      savecontent variable="mailBody"{
        WriteOutput("This message was sent by an automatic mailer built with
        cfmail:= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = ",
        & "<br><br>" & form.body);
      }
    /* set mail attributes using implicit setters provided */
    mailerService.setTo(form.mailto);
    mailerService.setFrom(form.mailFrom);
    mailerService.setSubject(form.subject);
    mailerService.setType("html");
    /* add mailparams */
    mailerService.addParam(file=expandpath(form.attachment),type="text/plain",remove=fal
```
se);

    / * send mail using send(). Attribute values specified in an end action like
    "send" will not persist after the action is performed */
    mailerService.send(body=mailBody);
    writeoutput("<h3>Thank you</h3> & "Thank you, " & mailfrom & "<br>" & "Your
    message," & subject & ", has been sent to " & mailto & ")
    }
}
</cfscript>
<p>
<form action = "mail1.cfm" method="POST">
<table>
  <tr>
    <td>TO</td>
    <td><input type = "Text" name = "MailTo"></td>
  </tr>
  <tr>
    <td>FROM</td>
    <td><input type = "Text" name = "MailFrom"></td>
  </tr>
  <tr>
    <td>SUBJECT</td>
    <td><input type = "Text" name = "Subject"></td>
  </tr>
  <tr>
    <td>ATTACHMENT</td>
    <td><input type = "file" name = "attachment"></td>
  </tr>
</table>
<br>
MESSAGE BODY:
<br>
<textarea name ="body" cols="40" rows="5" wrap="virtual"></textarea>
<!--- Establish required fields. --->
<input type = "hidden" name = "MailTo_required" value = "You must enter a
recipient">
<input type = "hidden" name = "MailFrom_required" value = "You must enter a
sender">
<input type = "hidden" name = "Subject_required" value = "You must enter a
subject">
<input type = "hidden" name = "Body_required" value = "You must enter some
text">
<input type = "hidden" name = "attachment_required" value = "You must select a
file">
<p><input type = "Submit" name = ""></p>
pdf

Description

Provides services to manipulate existing PDF documents in CFScript.

Syntax

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td>new pdf() or {createObject(&quot;component&quot;, &quot;pdf&quot;)}</td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>Any one of the following:</td>
</tr>
<tr>
<td></td>
<td>• pdfService=new pdf(attribute-value pair)</td>
</tr>
<tr>
<td></td>
<td>• pdfService.setAttributes(attribute-value pair)</td>
</tr>
<tr>
<td></td>
<td>• pdfService.setAttributeName(attribute_value_)</td>
</tr>
<tr>
<td></td>
<td>• pdfService.action_method(attribute-value pair)</td>
</tr>
<tr>
<td>Executing the service action</td>
<td>pdfService.action_method(attribute-value pair)</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>addQuads</th>
<th>algo</th>
<th>align</th>
<th>ascending</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottomMargin</td>
<td>compressTiffs</td>
<td>copyFrom</td>
<td>ddxfile</td>
</tr>
<tr>
<td>destination</td>
<td>directory</td>
<td>encodeAll</td>
<td>encrypt</td>
</tr>
<tr>
<td>flatten</td>
<td>foreground</td>
<td>format</td>
<td>height</td>
</tr>
<tr>
<td>hires</td>
<td>honourSpaces</td>
<td>hScale</td>
<td>image</td>
</tr>
<tr>
<td>imagePrefix</td>
<td>info</td>
<td>inputFiles</td>
<td>isBase64</td>
</tr>
<tr>
<td>jpgDpi</td>
<td>keepBookmark</td>
<td>leftMargin</td>
<td>maxBreadth</td>
</tr>
<tr>
<td>maxLength</td>
<td>maxScale</td>
<td>name</td>
<td>newOwnerPassword</td>
</tr>
<tr>
<td>newUserPassword</td>
<td>noAttachments</td>
<td>noBookmarks</td>
<td>noComments</td>
</tr>
<tr>
<td>noJavascripts</td>
<td>noLinks</td>
<td>noMetadata</td>
<td>noThumbnails</td>
</tr>
</tbody>
</table>
All attributes supported by the tag `<cfpdf>` can be used as attribute-value pairs. For example,

```
<cfpdf action="getInfo" source="myBook.pdf" name="PDFInfo">
```

can be used as

```
pdfInfo = pdfService.getPdfInfo(source="myBook.pdf", name="pdfInfo");
```

For details, see the Attributes section for the `<cfpdf>` tag.

**Methods**

- addParam

<table>
<thead>
<tr>
<th>Description</th>
<th>Used in CFScript to add <code>&lt;cfpdfparam&gt;</code> tags. Applicable only to action=&quot;merge&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td>pdfService.addParam(attribute-value pair)</td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by the <code>&lt;cfpdfparam&gt;</code> tag can be used as attribute-value pairs.</td>
</tr>
</tbody>
</table>

- The following PDF actions are available as methods. All these methods have similar arguments and syntax.

<table>
<thead>
<tr>
<th>addWatermark</th>
<th>removeWatermark</th>
<th>deletePages</th>
<th>getPDFInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>setPDFInfo</td>
<td>merge</td>
<td>processDDX</td>
<td>protect</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Returns</td>
<td>Syntax</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>read</td>
<td></td>
<td></td>
<td>serviceName.methodName(attribute-value pair)</td>
</tr>
<tr>
<td>write</td>
<td></td>
<td></td>
<td>pdfService.setAttributes (attribute-value pair)</td>
</tr>
<tr>
<td>thumbnail</td>
<td></td>
<td></td>
<td>pdfService.setAttributes (attribute-value pair)</td>
</tr>
<tr>
<td>transform</td>
<td></td>
<td></td>
<td>pdfService.setAttributes (attribute-value pair)</td>
</tr>
<tr>
<td>optimize</td>
<td></td>
<td></td>
<td>pdfService.setAttributes (attribute-value pair)</td>
</tr>
<tr>
<td>extractImage</td>
<td></td>
<td></td>
<td>pdfService.setAttributes (attribute-value pair)</td>
</tr>
<tr>
<td>extractText</td>
<td></td>
<td></td>
<td>pdfService.setAttributes (attribute-value pair)</td>
</tr>
<tr>
<td>addHeader</td>
<td></td>
<td></td>
<td>pdfService.setAttributes (attribute-value pair)</td>
</tr>
<tr>
<td>removeHeaderFooter</td>
<td></td>
<td></td>
<td>pdfService.setAttributes (attribute-value pair)</td>
</tr>
</tbody>
</table>

**Note**

In the list, `setPDFInfo` and `getPDFInfo` do not have identical actions in `cfpdf`. `cfpdf action="setinfo"` and `cfpdf action="getinfo"` represent them respectively.

Description

All methods correspond to the PDF actions specified for the tag `cfpdf`. For details of each method, refer to the corresponding section for `cfpdf`.

Returns

Depends on the action. If the `name` attribute is specified, the result of the pdf operation is returned. Else, an empty string. For example, the following code returns a structure containing the pdf information for "book.pdf":

```javascript
pdfinfo = pdfService.getPDFInfo(source="book.pdf", name="var"))
```

PDF manipulation is done using the `{{cfpdf tag. This is why, you must specify the `name` attribute. Accessing "var" directly does not work since "var" does not exist in the page variables scope.

Syntax

`serviceName.methodName(attribute-value pair)`

Arguments

All attributes supported by the `cfpdf` tag for a given action are supported.

- `setAttributes`
- `getAttributes`
## clearAttributes

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes all attributes added for the pdf function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>pdfService.clearAttributes(attribute_list)</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>A comma-separated list of attributes that must be removed.</td>
</tr>
</tbody>
</table>

## clearParams

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes params that were added using <code>addParam</code> method.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>pdfService.clearParams()</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
</tbody>
</table>

## clear

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes all attributes and params added using the <code>addParam</code> method.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>pdfService.clear()</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
</tbody>
</table>

### See also
**cfpdf**. Function summary

**History**

ColdFusion 9: Added this function.

**Usage**

This function corresponds to the *cfpdf* tag. For usage details, refer to the Usage section for *cfpdf*.

**Example**

```cfs
<h3>PDF Thumbnail</h3>
<cfscript>
   // Create a variable for the name of the PDF document.
   mypdf = "book";
   thumbnailsDirectory = ExpandPath("") & "\" & "#mypdf#_thumbnails";
   //create new PDF service
   pdfService = new pdf();
   //set attributes using implicit setters
   pdfService.setSource(expandpath('./#mypdf#.pdf'));
   //Use the getPdfInfo action to retrieve the total page count for the PDF document.
   PDFInfo = pdfService.getPdfInfo(name="pdfinfo");
   pageCount = PDFInfo.TotalPages;
   WriteOutput("pageCount=" & pageCount);
   //Generate a thumbnail image for each page in the PDF source document,
   //create a directory (if it does not exist) in the web root that is
   //a concatenation of the PDF source name and the word "thumbnails", and
   //save the thumbnail images in that directory.
   pdfService.thumbnail(destination=thumbnailsDirectory, scale=60, overwrite=true);
   //Loop through the images in the thumbnail directory and generate a link
   //from each image to the corresponding page in the PDF document.
   for(i="1";i lte pageCount;i++)
   {
      //Click the thumbnail image to navigate to the page in the PDF document.
      WriteOutput("<a href='#mypdf#.pdf##page=#i#' target='_blank'><img src='#mypdf#_thumbnails/#mypdf#_page_#i#.jpg'></a>");
   }
</cfscript>
```

#back to top
query

Description

Used to execute a query passing SQL statements to a data source using CFScript.

Syntax

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td>new query() or createObject(&quot;component&quot;, &quot;query&quot;)</td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>Any one of the following:</td>
</tr>
<tr>
<td></td>
<td>• queryService=new query(attribute-value_pair)</td>
</tr>
<tr>
<td></td>
<td>• queryService.setAttributes(attribute-value_pair)</td>
</tr>
<tr>
<td></td>
<td>• queryService.set_AttributeName_(attribute_value)</td>
</tr>
<tr>
<td></td>
<td>• queryService.execute(attribute-value_pair)</td>
</tr>
<tr>
<td>Executing the service action</td>
<td>queryService.execute(attribute-value_pair)</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>name</th>
<th>blockfactor</th>
<th>cachedafter</th>
<th>cachedwithin</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataSource</td>
<td>dbtype</td>
<td>debug</td>
<td>maxRows</td>
</tr>
<tr>
<td>password</td>
<td>result</td>
<td>timeout</td>
<td>username</td>
</tr>
<tr>
<td>sql</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All attributes supported by the tag cfquery can be used as attribute-value pairs. For example,

```xml
<cfquery Name="myName"> </cfquery>
```

can be used as

```javascript
queryService.setName("myName");
```
See also

cfquery, Function summary

History

ColdFusion 9: Added this function.

Methods

- **addParam**
  
<table>
<thead>
<tr>
<th>Description</th>
<th>Used in CFScript to add <code>cfqueryparam</code> tags to:</th>
</tr>
</thead>
</table>
  
  - Verify the data type of a query parameter
  - For DBMSs that support bind variables, to enable ColdFusion to use bind variables in the SQL statement
  
<table>
<thead>
<tr>
<th>Syntax</th>
<th><code>serviceName.addParam(attribute-value pair)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by <code>cfqueryparam</code> tag can be used as attribute-value pairs.</td>
</tr>
</tbody>
</table>

- **execute**
  
<table>
<thead>
<tr>
<th>Description</th>
<th>Used to execute SQL statements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>A component with the following properties set:</td>
</tr>
<tr>
<td>Result</td>
<td>For SQL queries that return a result set, for example, a &quot;SELECT&quot; SQL query.</td>
</tr>
<tr>
<td>Prefix</td>
<td>Equivalent to the <code>result</code> attribute for the <code>cfquery</code> tag.</td>
</tr>
</tbody>
</table>
  
<table>
<thead>
<tr>
<th>Syntax</th>
<th><code>queryService.execute(attribute-value pair)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments</td>
<td>All attributes supported by the <code>cfquery</code> tag.</td>
</tr>
</tbody>
</table>

- **setAttribute**
  
<table>
<thead>
<tr>
<th>Description</th>
<th>Sets attributes for the <code>query</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>queryService.setAttribute (attribute-value pair)</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by the <code>cfquery</code> tag.</td>
</tr>
</tbody>
</table>
### getAttributes

**Description**
Gets attributes that were set for the `query` function.

**Returns**
Returns a struct with all or some of the attribute values.

**Syntax**
```plaintext
queryService.get_Attributes_ (attributelist)
```

**Arguments**
A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

### clearAttributes

**Description**
Removes all attributes added for the `query` function.

**Returns**
Nothing

**Syntax**
```plaintext
queryService.clearAttributes(attribute_list)
```

**Arguments**
A comma-separated list of attributes.

### clearParams

**Description**
Removes queryparams that were added using the `addParam` method.

**Returns**
Nothing

**Syntax**
```plaintext
queryService.clearParams()
```

**Arguments**
None

### clear

**Description**
Removes all attributes and queryparams that were added using the `addParam` method.

**Returns**
Nothing

**Syntax**
```plaintext
queryService.clear()
```

**Arguments**
None

---

**Usage**
This function corresponds to the `cfquery` tag. For usage information, see Usage details for `cfquery`.

Example

```
<cfscript>
/*
This example shows how to create a query service in cfscript, set/get attributes
using implicit setters/getters, and also
how to execute the query and access the resultset
*/
param MaxRows="10";
param StartRow="1";
/*
Query database for information if cached database information has
not been updated in the last six hours; otherwise, use cached data.
*/
/* create a query service */
queryService = new query();
/* set properties using implicit setters */
queryService.setDatasource("cfdocexamples");
queryService.setName("GetParks");
queryService.setcachedwithin(CreateTimeSpan(0, 6, 0, 0));
/* Add sql queryparams using named and positional notation */
queryService.addParam(name="state",value="MD",cfsqltype="cf_sql_varchar");
queryService.addParam(value="National Capital Region",cfsqltype="cf_sql_varchar");
/* invoke execute() on the query object to execute the query and return a component
with properties result and prefix (which can be accessed as implicit getters) */
result = queryService.execute(sql="SELECT PARKNAME, REGION, STATE FROM Parks WHERE
STATE = :state and REGION = ? ORDER BY ParkName, State ");
GetParks = result.getResult();
/* getPrefix() returns information like recordcount,sql etc (typically whatever one
gets if one uses the result attribute of the cfquery tag */
metaInfo = result.getPrefix();
</cfscript>
<cfoutput>
<h4>Found #metaInfo.recordcount# records for '#metainfo.sqlparameters[2]#' in the
state '#metainfo.sqlparameters[1]#' </h4>
</cfoutput>
<--- Build HTML table to display query. ------------------------>
<table cellpadding="1" cellspacing="1">
<tr bgcolor="f0f0f0">
<td><b>Park Name</b></td>
<td><b>Region</b></td>
<td><b>State</b></td>
</tr>
</table>
</tr>
<tr bgcolor="f0f0f0">
<td>&nbsp;</td>
</tr>
</table>
</cfoutput>
```

<!--- Build HTML table to display query. ------------------------>
<table cellpadding="1" cellspacing="1">
<tr bgcolor="f0f0f0">
<td><b>Park Name</b></td>
<td><b>Region</b></td>
<td><b>State</b></td>
</tr>
</table>
</tr>
</table>

<!--- Output the query and define the startrow and maxrows parameters.
Use the query variable CurrentCount to keep track of the row you are displaying.-------->
<cfoutput query="GetParks" startrow="#StartRow#" maxrows="#MaxRows#">
  <tr>
    <td valign="top" bgcolor="ffffed">
      <b>#GetParks.CurrentRow#</b>
    </td>
    <td valign="top">
      <font size="-1">#ParkName#</font>
    </td>
    <td valign="top">
      <font size="-1">#Region#</font>
    </td>
    <td valign="top">
      <font size="-1">#State#</font>
    </td>
  </tr>
</cfoutput>

<!--- If the total number of records is less than or equal to the total number of rows, then offer a link to the same page, with the startrow value incremented by maxrows (in the case of this example, incremented by 10). --->
<tr><td colspan="4">
  <cfif (StartRow + MaxRows) LTE GetParks.RecordCount>
    <cfoutput><a href="#CGI.SCRIPT_NAME#?startrow=#Evaluate(StartRow + MaxRows)#">
      See next #MaxRows# rows</a></cfoutput>
  </cfif></td></tr>
**storedproc**

**Description**

Used to execute a stored procedure in a server database using CFScript. It specifies database connection information and identifies the stored procedure.

**Syntax**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td>new storedProc() or createObject(&quot;component&quot;, &quot;storedproc&quot;)</td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>Any one of the following:</td>
</tr>
<tr>
<td></td>
<td>• storedProcService=new storedproc(attribute-value_pair)</td>
</tr>
<tr>
<td></td>
<td>• storedProcService.setAttributes(attribute-value_pair)</td>
</tr>
<tr>
<td></td>
<td>• storedProcService.set_AttributeName_(attribute_value)</td>
</tr>
<tr>
<td></td>
<td>• storedProcService.execute(attribute-value_pair)</td>
</tr>
<tr>
<td>Executing the service action</td>
<td>storedProcService.execute(<em>attribute-value_pair</em>)</td>
</tr>
</tbody>
</table>

**Properties**

<table>
<thead>
<tr>
<th>datasource</th>
<th>procedure</th>
<th>debug</th>
<th>cachedafter</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachedwithin</td>
<td>blockfactor</td>
<td>password</td>
<td>result</td>
</tr>
<tr>
<td>returncode</td>
<td>username</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All attributes supported by the tag **cfstoredproc** are supported as attribute-value pairs. For example,

```html
cfstoredproc procedure= "sp_proc"
```

can be used as

```javascript
spService.setProcedure("sp_proc");
```

For details of the **cfstoredproc** tag attributes, see the Attributes section for **cfstoredproc**.
See also

cfstoredproc, Function summary

History

ColdFusion 9: Added this function.

Methods

• addParam

<table>
<thead>
<tr>
<th>Description</th>
<th>Used to add cfprocream tags.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>storedprocService.addParam(attribute-value pair)</td>
</tr>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by cfprocream tag can be used as attribute-value pairs.</td>
</tr>
</tbody>
</table>

• addProcResult

<table>
<thead>
<tr>
<th>Description</th>
<th>Used to add cfprocream tags to associate a query object with a result set returned by a stored procedure.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>storedprocService.addProcResult(attribute-value pair)</td>
</tr>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by the cfprocream tag can be used as attribute-value pairs.</td>
</tr>
</tbody>
</table>

• execute

<table>
<thead>
<tr>
<th>Description</th>
<th>Used to execute a stored procedure.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>A component on which the following methods can be invoked:</td>
</tr>
</tbody>
</table>

• getProcResultSets(): To access result sets returned by the procedure.
• getProcOutVariables(): To access OUT or INOUT variables returned by the procedure.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>storedprocService.execute(attribute-value pair)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments</td>
<td>All attributes supported by the cfstoredproc tag.</td>
</tr>
</tbody>
</table>
### setAttributes

<table>
<thead>
<tr>
<th>Description</th>
<th>Sets attributes for the <code>storedproc</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>storedProcService.setAttributes (attribute-value pair)</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by the <code>cfstoredproc</code> tag.</td>
</tr>
</tbody>
</table>

### getAttributes

<table>
<thead>
<tr>
<th>Description</th>
<th>Gets attributes that were set for the <code>storedproc</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Returns a struct with all or some of the attribute values.</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>storedProcService.get_Attributes_ (attributelist)</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>A comma-separated list of attributes. If no list is specified, all defined attributes are returned.</td>
</tr>
</tbody>
</table>

### clearAttributes

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes all attributes added for the <code>storedProc</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>storedProcService.clearAttributes(attribute_list)</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>A comma-separated list of attributes.</td>
</tr>
</tbody>
</table>

### clearParams

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes <code>cfprocparam</code> tags added using the <code>add Param</code> method.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>storedProcService.clearParams()</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>clearProcResults</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>Returns</td>
</tr>
<tr>
<td></td>
<td>Syntax</td>
</tr>
<tr>
<td></td>
<td>Arguments</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>clear</th>
<th>Description</th>
<th>Removes all attributes and params that were added using the methods <code>addProcResults</code> and <code>addParam</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td></td>
<td>Syntax</td>
<td><code>storedProcService.clear()</code></td>
</tr>
<tr>
<td></td>
<td>Arguments</td>
<td>None</td>
</tr>
</tbody>
</table>

**Usage**

This function corresponds to the `cfstoredproc` tag. For usage details, refer to the Usage section for `cfstoredproc`.

**Example**
<cfscript>
//If submitting a new book, insert the record and display confirmation
if(isDefined("form.title"))
{
    //create a new storedproc service
    spService = new storedproc();
    //set attributes using implicit setters
    spService.setDatasource("books");
    spService.setProcedure("Insert_Book");
    //add procparams using addParam
    spService.addParam(cfsqltype="cf_sql_varchar", type="in", value=form.title);
    spService.addParam(cfsqltype="cf_sql_numeric", type="in", value=form.price);
    spService.addParam(cfsqltype="cf_sql_date", type="in", value=form.publishDate);
    spService.addParam(cfsqltype="cf_sql_numeric", type="out", variable="bookId");
    //add procresults using addProcResult
    spService.addProcResult(name="rs1", resultset=1);
    //execute the stored procedure
    result = spService.execute();
    //getprocOutVariables() returns any OUT or INOUT variables added using addParams()
    bookId = result.getprocOutVariables().bookId;
    //getProcResultSets() returns resultsets added using addProcresult()
    listOfBooks = result.getProcResultSets().rs1;
    WriteOutput("<h3>List of Books</h3>");
    writeDump(listOfBooks);
    //output data
    WriteOutput("<h3>" & "'" & form.title & "'" & " inserted into database. The ID is " & bookId & "</h3>");
}
</cfscript>
<cfform action="#CGI.SCRIPT_NAME#" method="post">
    <h3>Insert a new book</h3>
    <table>
        <tr>
            <td>Title:</td>
            <td><cfinput type="text" size="20" required="yes" name="title"/></td>
        </tr>
        <tr>
            <td>Price:</td>
            <td><cfinput type="text" size="20" required="yes" name="price" validate="float"/></td>
        </tr>
        <tr>
            <td>Publish Date:</td>
            <td><cfinput type="datefield" name="publishdate" mask="mm/dd/yyyy" size="20"/></td>
        </tr>
    </table>
    <input type="submit" value="Insert Book"/>
</cfform>
Script functions implemented as CFCs in ColdFusion 9 Update 1

Function summary

The following table lists the script functions and the equivalent ColdFusion tag.

<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent ColdFusion Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbinfo</td>
<td>cfdbinfo</td>
</tr>
<tr>
<td>imap</td>
<td>cfimap</td>
</tr>
<tr>
<td>pop</td>
<td>cfpop</td>
</tr>
<tr>
<td>ldap</td>
<td>cfldap</td>
</tr>
<tr>
<td>feed</td>
<td>cffeed</td>
</tr>
</tbody>
</table>

**dbinfo**

*Description*

Used in CFScript to retrieve information about a data source such as database details, tables, queries, procedures, foreign keys, indexes, and version information about the database, driver, and JDBC.

*Syntax*

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td>new dbinfo() or createObject(&quot;component&quot;, &quot;dbinfo&quot;);</td>
</tr>
<tr>
<td>Executing the service action</td>
<td>dbinfoService.action_method(attribute-value_pair);</td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>See Initializing the attributes below.</td>
</tr>
<tr>
<td>Getting the CFC properties</td>
<td>See Getting the CFC Properties below.</td>
</tr>
<tr>
<td>Working with the data returned</td>
<td>data=dbinfoService.action_method(attribute-value_pair);writedump(data);</td>
</tr>
</tbody>
</table>

*Properties*

<table>
<thead>
<tr>
<th>datasource</th>
<th>dbname</th>
<th>name</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern</td>
<td>table</td>
<td>username</td>
<td></td>
</tr>
</tbody>
</table>
All attributes supported by the tag `cfdbinfo` can be used as attribute-value pairs. For example,

```xml
<cfdbinfo userName="myUserName">
```

can be used as

```java
dbinfoService.setUserName("myUserName");
```

For details, see the Attributes section for the `cfdbinfo` tag.

See also

Function summary

History

ColdFusion 9.0.1: Added this function.

Methods

The following `dbinfo` types are available as methods. All methods have similar arguments and syntax.

<table>
<thead>
<tr>
<th>dbnames</th>
<th>tables</th>
<th>columns</th>
<th>version</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedures</td>
<td>foriegnkeys</td>
<td>index</td>
<td></td>
</tr>
</tbody>
</table>

Description

All methods correspond to the type of information supported by the tag `cfdbinfo`. For details of each method, see the relevant section for the tag `cfdbinfo` in *ColdFusion 9 CFML Reference*.

Returns

All methods return a query object.

Syntax

```java
dbinfoService.methodName(attribute-value pair);
```

Arguments

All attributes supported by the tag `cfdbinfo`.

**setAttributes, getAttributes, clearAttributes, clear, setProperties, getProperties, clearProperties, and clearProperties.** For details, see *Methods common to all functions*.

Usage

This function corresponds to the tag `cfdbinfo`. For details, see the Usage section for the tag `cfdbinfo`.

Example
<cfscript>
    d = new dbinfo(datasource=" cfartgallery ").dbnames(datasource="ajax");
    writedump(d);
    d = new dbinfo(datasource=" ajax").dbnames();
    writedump(d);
</cfscript>

imap

Description

Used in CFScript to query an IMAP server to retrieve and manage mails within multiple folders.

Syntax

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td>new imap(); or createObject(&quot;component&quot;, &quot;imap&quot;);</td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>See Initializing the attributes below.</td>
</tr>
<tr>
<td>Executing the service action</td>
<td>imapService.methodName(<em>attribute-value_pair</em>)</td>
</tr>
<tr>
<td>Getting the CFC properties</td>
<td>See Getting the CFC Properties below.</td>
</tr>
<tr>
<td>Working with returned data</td>
<td>imapResult=imapService.action_method(<em>attribute-value_pair</em>);</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>attachmentpath</th>
<th>connection</th>
<th>folder</th>
<th>generateuniquefilenames</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxrows</td>
<td>messagenumber</td>
<td>name</td>
<td>newfolder</td>
</tr>
<tr>
<td>password</td>
<td>port</td>
<td>recurse</td>
<td>secure</td>
</tr>
<tr>
<td>server</td>
<td>startrow</td>
<td>stoponerror</td>
<td>timeout</td>
</tr>
<tr>
<td>uid</td>
<td>username</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All attributes supported by the tag cfimap can be used as attribute-value pairs. For example,
<cfimap action="open" connection = "myconnection">

can be used as

```coldfusion
imapService = new imap(server="myimapserver", username="myusername", password="mypassword", port="myport", secure="yes");
imapService.open();
```

⚠️ **Note**

If connection properties such as server, username, password, port, and secure are specified either during initialization or when open method is called, a connection is created implicitly. Therefore, you need not specify the properties for further actions. If sandbox security is turned on, the directory referred to by the property `attachmentPath` must be given the required permission. By default, the temp directory is used.

For details of the attributes, see the Attributes section for the tag `cfimap`.

**See also**

**Function summary**

**History**

ColdFusion 9.0.1: Added this function.

**Methods**

The following imap actions are available as methods. All methods have similar arguments and syntax.

<table>
<thead>
<tr>
<th>getall</th>
<th>delete</th>
<th>open</th>
<th>close</th>
</tr>
</thead>
<tbody>
<tr>
<td>markRead</td>
<td>createFolder</td>
<td>deleteFolder</td>
<td>renameFolder</td>
</tr>
<tr>
<td>listAllFolders</td>
<td>moveMail</td>
<td>getHeaderOnly</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

All methods correspond to the type of information supported by the tag `cfimap`. For details of each method, see the relevant section of `cfimap` in the ColdFusion 9 CFML Reference.

**Returns**

A query object for methods `getAll`, `getHeaderOnly`, and `listAllFolders`. Else, nothing.
Syntax

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td>new pop(); or createObject(&quot;component&quot;, &quot;pop&quot;);</td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>See Initializing the attributes below.</td>
</tr>
<tr>
<td>Executing the service action</td>
<td>popService.action_method(<em>attribute-value pair</em>);</td>
</tr>
<tr>
<td>Getting the CFC properties</td>
<td>See Getting the CFC Properties below.</td>
</tr>
</tbody>
</table>

Arguments

- setAttributes, getAttributes, clearAttributes, clear, setProperties, getProperties, and clearProperties. For details, see Methods common to all functions.

Usage

This function corresponds to the tag cfimap. See the Usage section for cfimap in the ColdFusion 9 CFML Reference for details.

Example

```cfscript
<cfscript>

m = new imap();
m.setAttributes(server="#REQUEST.server#", username="#REQUEST.username#",
    password="#REQUEST.password#", secure="#REQUEST.secure#",
    connection="#REQUEST.connectionname#", stoponerror="#REQUEST.stoponerror#");
m.open();
master = m.getAll(connection = "+REQUEST.connectionname#", name = "queryname",
    stoponerror = "+REQUEST.stoponerror#");
writedump(master);

</cfscript>
```

pop

Description

Used in CFScript to retrieve or delete e-mail messages from a POP mail server.

Syntax

```
Working with data returned

popresult = popService.action_method (attribute-value pair); where popresult is a query object if the action_method is getAll or getHeaderOnly. For any other method, nothing is returned.

Properties

<table>
<thead>
<tr>
<th>server</th>
<th>attachmentPath</th>
<th>debug</th>
</tr>
</thead>
<tbody>
<tr>
<td>generateUniqueFileNames</td>
<td>maxRows</td>
<td>messageNumber</td>
</tr>
<tr>
<td>password</td>
<td>port</td>
<td>startRow</td>
</tr>
<tr>
<td>uid</td>
<td>username</td>
<td>timeout</td>
</tr>
</tbody>
</table>

All attributes supported by the tag cfpop can be used as attribute-value pairs. For example,

```html
<cfpop server="#form.popserver# " action = "getHeaderOnly" name = "GetHeaders">  
```

can be used as

```javascript
popHeaders = popService.getHeaderOnly(server="#form.popserver#");  
```

⚠️ Note

name is a required attribute in cfpop, but not in CFScript.

See also

Function summary

History

ColdFusion 9.0.1: Added this function.

Methods

The following pop actions are available as methods. All methods have similar arguments and syntax.

- getHeaderOnly
- getAll
- delete
• Description
All methods correspond to the type of information supported by the tag `cfpop`. For details of each method, see the relevant section of `cfpop` in the *ColdFusion 9 CFML Reference*.

<table>
<thead>
<tr>
<th>Returns</th>
<th>All methods except <code>delete</code> returns a query object.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>popService.methodName(attribute-value pair)</code></td>
</tr>
</tbody>
</table>

| Arguments | All attributes supported by the tag `cfpop`. |

• `setAttributes, getAttributes, clearAttributes, clear, setProperties, getProperties, and clearProperties`. For details, see Methods common to all functions.

**Usage**

This function corresponds to the tag `cfpop`. For usage details, see the Usage section for `cfpop`.

**Example**

```cfscript
<cfscript>
    p = createObject("component","pop");
    p.setAttributes(server="#popServer#",username="failoveruser",password="#popPassword #");
    r = p.GetAll(name="results",maxRows = "2");
    writeoutput("getAll Passed<br>");
    r = p.GetAll(messageNumber = "2");
    writeoutput("#r.FROM# & "<br>");

    r= p.GETHEADERONLY(messageNumber = "1");
    writeoutput("#r.subject# & "<br>");
</cfscript>
```

**ldap**

**Description**

Used in CFScript to provide an interface to a Lightweight Directory Access Protocol (LDAP) directory server, such as the Netscape Directory Server.

**Syntax**
Creating the service

new ldap(); or createObject("component", "ldap");

Initializing the attributes

See Initializing the attributes below.

Executing the service action

ldapService.action_method(attribute-value pair);

Getting the CFC properties

See Getting the CFC Properties below.

Working with data

ldapresult = ldapService.query(attribute-value pair). For other methods, nothing is returned.

Properties

<table>
<thead>
<tr>
<th>server</th>
<th>attributes</th>
<th>delimiter</th>
</tr>
</thead>
<tbody>
<tr>
<td>dn</td>
<td>filter</td>
<td>maxRows</td>
</tr>
<tr>
<td>name</td>
<td>password</td>
<td>port</td>
</tr>
<tr>
<td>referral</td>
<td>returnAsBinary</td>
<td>scope</td>
</tr>
<tr>
<td>separator</td>
<td>sort</td>
<td>sortcontrol</td>
</tr>
<tr>
<td>startRow</td>
<td>timeout</td>
<td>userName</td>
</tr>
</tbody>
</table>

All attributes supported by the tag cfldap can be used as attribute-value pairs. For example,

```
<cfldap action="add" server="ldap.uconn.edu">
```
can be used as

```
ldapService.add(server="ldap.uconn.edu");
```

For details, see the Attributes section for the tag cfldap.

Methods

The following ldap actions are available as methods. All methods have similar arguments and syntax.

- query
- add
- modify
- modifyDn
- delete
### Description
All methods correspond to the actions supported by the tag `cfldap`. For details of each method, see the relevant section of `cfldap` in the *ColdFusion 9 CFML Reference*.

### Returns
If method is `query`, returns a query object. Else, none.

### Syntax
`ldapService.methodName(attribute-value)`

### Arguments
All attributes supported by the tag `cfldap`.

### setAttributes
For details, see Methods common to all functions

### getAttributes, clearAttributes, clear, setProperties, getProperties, and clearProperties
For details, see Methods common to all functions.

### setLdapAttributes

| Description | Sets the property `attributes`. |
| Returns     | Nothing                         |
| Syntax      | `ldapService.setLdapAttributes(attribute-value)`; |
| Arguments   | A string that contains the value of the property `attributes`. |

### getLdapAttributes

| Description | Gets the property `attributes`. |
| Returns     | A string that contains the value of the property `attributes`. |
| Syntax      | `myattributes = ldapService.getLdapAttributes(__)`; |

**See also**

**Function summary**

**History**

ColdFusion 9.0.1: Added this function.

**Usage**

This function corresponds to the tag `cfldap`. For usage details, see the Usage section for `cfldap`. 
Example

```
<cfscript>
  l = new ldap();
l.setLdapAttributes("objectclass=top, person, organizationalPerson,
inetOrgPerson;cn=Joe Smith; sn=Smith; mail=spenella@allaire.com;
telephonenumber=(617) 761 - 2128");
l.setUsername("uid=admin,ou=system");
l.setPassword("administrator");
l.setPort(port);
l.setServer(ldapserver);
l.setdn("ou=People+o=aribus.com,dc=example,dc=com");

l.add();
l.clearAttributes(); result = l.query(name="apache",
  attributes="dn,cn,o,ou,c,mail,telephonenumber",
  start="dc=example,dc=com",
  scope="SUBTREE",
  filter="(&cn=Joe Smith)(ou=people))");

writeoutput("<b>Adding and Querying a LDAP entry : </b>" & "CN = " & result.CN & "
  DN = " & result.DN & "<br> ");
l.clearAttributes();
l.delete(
  DN="ou=People+o=aribus.com,dc=example,dc=com",
);

</cfscript>
```

**feed**

**Description**

Used in CFScript to read or create an RSS or Atom syndication feed. This service reads RSS versions 0.90, 0.91, 0.92, 0.93, 0.94, 1.0, and 2.0, and Atom 0.3 or 1.0. It can create RSS 2.0 or Atom 1.0 feeds.

**Syntax**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the service</td>
<td><code>new feed() or createObject(&quot;component&quot; &quot;feed&quot;)</code></td>
</tr>
<tr>
<td>Initializing the attributes</td>
<td>See Initializing the attributes below.</td>
</tr>
<tr>
<td>Executing the service action</td>
<td><code>feedService.action_method(attribute-value_pair)</code></td>
</tr>
</tbody>
</table>
### Getting the CFC properties
See *Getting the CFC Properties* below.

### Working with the data returned
- `feedresult = feedService.read(attribute-value_pair)` where `feedresult` is a struct with the keys `name`, `query`, `properties`, and `xmlvar`.
- `feedresult = feedService.create(attribute-value_pair)` where `feedresult` is a string that contains the `xmlvar`.

### Properties

<table>
<thead>
<tr>
<th>columnMap</th>
<th>enclosureDir</th>
<th>escapeChar</th>
<th>ignoreEnclosureError</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (optional in CFScript)</td>
<td>outputFile</td>
<td>overwrite</td>
<td>overwriteEnclosure</td>
</tr>
<tr>
<td>properties (optional in CFScript)</td>
<td>proxyPassword</td>
<td>proxyPort</td>
<td>proxyServer</td>
</tr>
<tr>
<td>proxyUser</td>
<td>query (optional in CFScript)</td>
<td>source</td>
<td>timeout</td>
</tr>
<tr>
<td>userAgent</td>
<td>xmlvar (optional in CFScript)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All attributes supported by the tag `cffeed` can be used as attribute-value pairs. For example,

```xml
<cffeed action="read" source="http://googleblog.blogspot.com/atom.xml" query="feedQuery" properties="feedMetadata" />
```

can be used as

```coldfusion
feedservice.read(source="http://googleblog.blogspot.com/atom.xml", query="feedQuery", properties="feedMetadata");
```

### See also

**Function summary**

**History**

ColdFusion 9.0.1: Added this function.

**Methods**
### create

**Description**
Creates an RSS 2.0 or Atom 1.0 feed XML document and saves it in a variable, writes it to a file, or both.

**Returns**
String representing the xmlvar

**Syntax**
```plaintext
feedService.create (attribute-value pair);
```

**Arguments**
All attributes supported by the tag `cffeed`.

### read

**Description**
 Parses an RSS or Atom feed from a URL or an XML file and saves it in a structure or query. You can also get feed metadata in a separate structure.

**Returns**
Struct with the following keys:

- name
- query
- properties
- xmlvar

**Syntax**
```plaintext
feedService.read (attribute-value pair);
```

**Arguments**
All attributes supported by the tag `cffeed`.

### setAttributes, getAttributes, clearAttributes, clear, setProperties, getProperties, and clearProperties. For details, see Methods common to all functions

### getFeedProperties

**Description**
Returns the value of the property `properties`.

**Returns**
Struct or error (if property is not set)

**Syntax**
```plaintext
feedService.getFeedProperties()
```

**Arguments**
None

### setFeedProperties

**Description**
Sets the value of the property `properties`.

**Returns**
Nothing
Syntax

```
feedService.setFeedProperties()
```

Arguments

```
properties struct
```

Usage

This service corresponds to the tag `cffeed`. For usage, see Usage section for `cffeed`.

Example

```
<cfscript>
  f = new feed();
  r = f.read(source=feedpath);

  writeoutput("Name : " & r.name.title & "<br>");
  writeoutput("Properties : " & r.properties.version & "<br>");
  writeoutput("Query : " & r.query.recordcount & "<br>");
  writeoutput("XMLVar : " & r.xmlvar.length() & "<br>");
</cfscript>
```

Methods common to all functions

The following methods are common to all script functions:

- **setAttributes**

  Description | Sets attributes for the function.
  Returns | Nothing
  Syntax | `service_name.setAttributes (attribute-value pair);`
  Arguments | All attributes supported by the equivalent tag.

- **getAttributes**

  Description | Gets the attributes set for the function.
  Returns | Returns a struct with all or some attribute values.
  Syntax | `service_name.get_Attributes_ (attribute-list);`
  Arguments | A comma-separated list of attributes. If no list is specified, all defined attributes are returned.

- **clearAttributes**
<table>
<thead>
<tr>
<th>Description</th>
<th>Removes all attributes added for the function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>_service_name.clearAttributes(_attribute_list);</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>A comma-separated list of attributes.</td>
</tr>
</tbody>
</table>

- **clear**

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes all attributes added for the function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>service_name.clear();</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
</tbody>
</table>

- **clearProperties**

<table>
<thead>
<tr>
<th>Description</th>
<th>Removes all properties added for the function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>service_name.clearProperties(attribute_list);</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>If nothing is specified, all properties are cleared.</td>
</tr>
</tbody>
</table>

- **setProperties**

<table>
<thead>
<tr>
<th>Description</th>
<th>Sets properties for the function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Nothing</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>service_name.setproperties(attribute-value pair);</code></td>
</tr>
<tr>
<td>Arguments</td>
<td>All attributes supported by the equivalent tag.</td>
</tr>
</tbody>
</table>

- **getProperties**

<table>
<thead>
<tr>
<th>Description</th>
<th>Gets the properties set for the function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Returns a struct with all or some attribute values.</td>
</tr>
</tbody>
</table>
### Initializing the attributes

You can initialize the attributes using one of the following ways:

- `service_name=new dbinfo(attribute-value pair)`
- `service_name=new dbinfo().init(attribute-value pair)`
- `service_name.setAttributes(_attribute-value pair_)`
- `service_name.setAttributeName_(attribute_value)`
- `service_name.action_method(attribute-value_pair)`
- `service_name.setProperties (attribute_value)`

### Getting the CFC Properties

Get the CFC properties using one of the following ways:

- `service_name.getAttributeName(attributeList)`
- `service_name.getProperties (attributeList)`
- `service_name.getAttributes (attributeList)`

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>service_name.getProperties (attributelist);</code></td>
<td>A comma-separated list of attributes. If no list is specified, all defined attributes are returned.</td>
</tr>
</tbody>
</table>
Example

ldap
  Description
  Syntax
  Properties
  Methods
  See also
  History
  Usage
  Example

feed
  Description
  Syntax
  Properties
  Methods
  Usage
  Example

Methods common to all functions
  Initializing the attributes
  Getting the CFC Properties
ColdFusion Flash Form Style Reference

You can specify styles in ColdFusion forms tags when you display the form or form element in Flash format.

⚠️ **Note**

The column labeled *inh* indicates whether a style is inherited by child controls, such as the form controls in a vbox.

- **Styles valid for all controls**
- **Styles for cfform**
- **Styles for cfformgroup with horizontal or vertical type attributes**
- **Styles for box-style cfformgroup elements**
- **Styles for cfformgroup with accordion type attribute**
- **Styles for cfformgroup with tabnavigator type attribute**
- **Styles for cfformitem with hrule or vrule type attributes**
- **Styles for cfformitem with radio, checkbox, button, image, or submit type attributes**
- **Styles for cfinput with text, password, or hidden type attributes**
- **Styles for cftextarea tag and cfinput with text, password, or hidden type attributes**
- **Styles for cfselect with size attribute value of 1**
- **Styles for cfselect with size attribute value greater than 1**
- **Styles for cfcalendar tag and cfinput with dateField type attribute**
- **Styles for the cfgrid tag**
- **Styles for the cftree tag**

#back to top
## Styles valid for all controls

The following styles are valid for all ColdFusion Flash format form tags except for `cfformitem` tags with the following `type` attributes, which do not take `style` attributes:

- `html`
- `space`

These styles do not cause errors when used in all other tags. However, many styles do not have any effect when used in some tags.

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backgroundAlpha</td>
<td>N</td>
<td>Alpha (transparency) level of the SWF file or image defined by <code>backgroundImage</code>. Valid values range from 0 (transparent) to 100 (opaque). The default value is 100.</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>Y</td>
<td>Format: color; background color of the control. Has no effect if specified in a <code>cformControl</code> tag, which uses the <code>background-color</code> style to control the color. Also ignored by <code>cformInput</code> tags of <code>type button</code>, <code>img</code>, <code>submit</code>, <code>radio</code>, and <code>checkbox</code>, because they are filled with the button face or other graphics.</td>
</tr>
<tr>
<td>backgroundDisabledColor</td>
<td>Y</td>
<td>Format: color; background color of components when disabled. The default value is <code>##EFEFFF</code> (light gray).</td>
</tr>
<tr>
<td>backgroundSize</td>
<td>N</td>
<td>Scales the image specified by <code>backgroundImage</code> to different percentage sizes. By default, the value is <code>auto</code>, which maintains the original size of the image. A value of 100% stretches the image to fit the entire screen. Include the percent sign with the value.</td>
</tr>
<tr>
<td>barColor</td>
<td>Y</td>
<td>Format: color; color of the outer bar.</td>
</tr>
<tr>
<td>borderCapColor</td>
<td>Y</td>
<td>Format: color; outside left and outside right color for skins.</td>
</tr>
<tr>
<td>Property</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>borderColor</td>
<td>Y</td>
<td>Format: color; black section of a three-dimensional border or the color section of a two-dimensional border.</td>
</tr>
<tr>
<td>borderSides</td>
<td>N</td>
<td>Bounding box sides. Only used when borderStyle=&quot;solid&quot;. Space-delimited string containing the sides of the border to show. Order is not important. The default value is &quot;left top right bottom&quot;.</td>
</tr>
<tr>
<td>borderStyle</td>
<td>Y</td>
<td>Bounding box style. The possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• inset (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• none</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• outset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• solid</td>
</tr>
<tr>
<td>borderThickness</td>
<td>N</td>
<td>Bounding box thickness. Only used when borderStyle=&quot;solid&quot;. The default value is 1.</td>
</tr>
<tr>
<td>color</td>
<td>Y</td>
<td>Format: color; text color of a component's label.</td>
</tr>
<tr>
<td>cornerRadius</td>
<td>N</td>
<td>Radius of component corners. The default value is 0.</td>
</tr>
<tr>
<td>disabledColor</td>
<td>Y</td>
<td>Format: color; color of the component if it is disabled.</td>
</tr>
<tr>
<td>dropShadow</td>
<td>N</td>
<td>Format: Boolean; controls the visibility of the component's drop shadow. The default value is false. This style must be used with borderStyle=&quot;solid&quot;. For drop shadows to appear on containers, set backgroundColor or backgroundImage. Otherwise, since the default background of a container is transparent, the shadow appears behind the container.</td>
</tr>
<tr>
<td>errorColor</td>
<td>Y</td>
<td>Format: color; color of the error text.</td>
</tr>
<tr>
<td>Property</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>fillColors</td>
<td>N</td>
<td>Format: color; colors used to tint the background of the control. Pass the same color for both values for &quot;flat&quot; looking control. The default value is #E6EEE,#FFFEFF.</td>
</tr>
<tr>
<td>fontFamily</td>
<td>Y</td>
<td>Comma-separated list of fonts to use, in descending order of desirability. You can use any font family name. If you specify a generic font name, it is converted to an appropriate device font. Flash can only use fonts that are installed on the client system.</td>
</tr>
<tr>
<td>fontSize</td>
<td>Y</td>
<td>Format: length; size of the text.</td>
</tr>
<tr>
<td>fontStyle</td>
<td>Y</td>
<td>Determines whether the text is italic. Recognized values are normal and italic. The default value is normal.</td>
</tr>
<tr>
<td>fontWeight</td>
<td>Y</td>
<td>Determines whether the text is bold. Recognized values are normal and bold. The default value is normal.</td>
</tr>
<tr>
<td>highlightColor</td>
<td>Y</td>
<td>Format: color; color of the control when it is in focus.</td>
</tr>
<tr>
<td>horizontalGap</td>
<td>N</td>
<td>Format: length; number of pixels between children in the horizontal direction.</td>
</tr>
<tr>
<td>leading</td>
<td>N</td>
<td>Additional vertical space between lines of text. The default value is no leading.</td>
</tr>
<tr>
<td>marginLeft</td>
<td>N</td>
<td>Format: length; number of pixels between the container's left border and its content area.</td>
</tr>
<tr>
<td>marginRight</td>
<td>N</td>
<td>Format: length; number of pixels between the container's right border and its content area.</td>
</tr>
<tr>
<td>scrollTrackColor</td>
<td>Y</td>
<td>Format: color; scroll track for a scroll bar. The default value is #EFEFFEE (light gray).</td>
</tr>
<tr>
<td>Property</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>selectedFillColors</td>
<td>N</td>
<td>Format: colors; two colors used to tint the background of the control when in its selected state. Pass the same color for both values for &quot;flat&quot; looking control. The default value is undefined, which means the colors are derived from themeColor.</td>
</tr>
<tr>
<td>textAlign</td>
<td>Y</td>
<td>Aligns text in a container. Recognized values are left, right, and center. The default value is right.</td>
</tr>
<tr>
<td>textDecoration</td>
<td>N</td>
<td>Determines whether the text is underlined or not. Recognized values are none and underline. The default value is none.</td>
</tr>
<tr>
<td>textIndent</td>
<td>Y</td>
<td>Format: length; offset of first line of text from the left side of the container. The default value is 0.</td>
</tr>
</tbody>
</table>
| themeColor           | Y        | Format: color; background color of a component. The possible values are:  
  - haloGreen  
  - haloBlue  
  - haloOrange  
  - haloSilver |
| verticalGap          | N        | Format: length; number of pixels between children in the vertical direction. |
### Styles for cfForm

The following styles apply to the `cfForm` tag:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background-color</td>
<td></td>
<td>Format: color; background color of the form.</td>
</tr>
<tr>
<td>indicatorGap</td>
<td>Y</td>
<td>Format: length; number of pixels between the label and child components. The default value is 14.</td>
</tr>
<tr>
<td>labelWidth</td>
<td>Y</td>
<td>Format: length; width of the form labels. The default value is the length of the longest label in the form.</td>
</tr>
<tr>
<td>marginBottom</td>
<td>N</td>
<td>Format: length; number of pixels between the container's bottom border and its content area. The default value is 16.</td>
</tr>
<tr>
<td>marginTop</td>
<td>N</td>
<td>Format: length; number of pixels between the container's top border and its content area. The default value is 16.</td>
</tr>
<tr>
<td>verticalGap</td>
<td>N</td>
<td>Format: length; number of pixels between children in the vertical direction. The default value is 8.</td>
</tr>
</tbody>
</table>
## Styles for cfformgroup with horizontal or vertical type attributes

The following styles apply to the **cfformgroup** tag with **type** attributes `horizontal` or `vertical`:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>horizontalAlign</td>
<td>N</td>
<td>Horizontal alignment of children. Possible values are left, center, and right. The default value is left.</td>
</tr>
<tr>
<td>horizontalGap</td>
<td>N</td>
<td>Format: length; number of pixels between children in the horizontal direction. The default value is 6.</td>
</tr>
<tr>
<td>indicatorGap</td>
<td>Y</td>
<td>Format: length; number of pixels between the label and child components. The default value is 14.</td>
</tr>
<tr>
<td>labelWidth</td>
<td>Y</td>
<td>Format: length; width of the form labels. The default value is the length of the longest label in the form.</td>
</tr>
<tr>
<td>marginBottom</td>
<td>N</td>
<td>Format: length; number of pixels between the container’s bottom border and its content area. The default value is 0.</td>
</tr>
<tr>
<td>marginTop</td>
<td>N</td>
<td>Format: length; number of pixels between the container’s top border and its content area. The default value is 0.</td>
</tr>
<tr>
<td>verticalGap</td>
<td>N</td>
<td>Format: length; number of pixels between children in the vertical direction. The default value is 6.</td>
</tr>
</tbody>
</table>
Styles for box-style cfformgroup elements

The following styles apply to the cfformgroup tag with the following type attributes. Some types have additional attributes, which are listed in the following sections.

- hbox
- vbox
- hdividedbox
- vdividedbox
- panel
- tile
- page

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>horizontalAlign</td>
<td>N</td>
<td>Horizontal alignment of children in the container. The default value is left. Possible values are left, center, and right.</td>
</tr>
<tr>
<td>horizontalGap</td>
<td>N</td>
<td>Format: length; number of pixels between children in the horizontal direction. The default value is 8 (6 for a tile container).</td>
</tr>
<tr>
<td>marginBottom</td>
<td>N</td>
<td>Format: length; number of pixels between the container's bottom border and its content area. The default value is 0.</td>
</tr>
<tr>
<td>marginTop</td>
<td>N</td>
<td>Format: length; number of pixels between the container's top border and its content area. The default value is 0.</td>
</tr>
<tr>
<td>verticalAlign</td>
<td>N</td>
<td>Vertical alignment of children in the container. The default value is top. Possible values are top, middle, and bottom.</td>
</tr>
<tr>
<td>verticalGap</td>
<td>N</td>
<td>Format: length; number of pixels between children in the vertical direction. The default value is 8 (6 for a tile container).</td>
</tr>
</tbody>
</table>

Styles specific to cfformgroup with hdividedbox or vdividedbox type attributes

The following additional styles apply to the cfformgroup tag with type="hdividedbox", or type="vdividedbox":

© 2014 Adobe Systems Incorporated. All rights reserved.
<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dividerAffordance</td>
<td>N</td>
<td>Format: length; width (hdividedbox) or height (vdividedbox) in pixels of the area of the divider that the user can select with the mouse pointer. The default value is 6.</td>
</tr>
<tr>
<td>dividerColor</td>
<td>Y</td>
<td>Format: color; color of the dividers in their up state. The default value is #AAAAAA.</td>
</tr>
<tr>
<td>dividerThickness</td>
<td>N</td>
<td>Format: length; thickness in pixels of the dividers. The default value is 4.</td>
</tr>
</tbody>
</table>

**Styles specific to cfformgroup with panel type attribute**

The following additional styles apply to the `cfformgroup` tag with `type="panel"`:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cornerRadius</td>
<td>N</td>
<td>Format: length; radius of corners of the window frame. The default value is 8.</td>
</tr>
<tr>
<td>dropShadow</td>
<td>N</td>
<td>Boolean value specifying whether the panel has a drop shadow. The default value is true.</td>
</tr>
<tr>
<td>footerColors</td>
<td>Y</td>
<td>Format: color; comma-delimited list of two colors used to draw the footer (ControlBar) background. The first color is the top color. The second color is the bottom color. The default value is ##F4F5F7,{{ ##E1E5EB}}.</td>
</tr>
<tr>
<td>headerColors</td>
<td>Y</td>
<td>Format: color; comma-delimited list of two colors used to draw the header. The first color is the top color. The second color is the bottom color. The default value is # #E1E5EB, ##F4F5F7.</td>
</tr>
<tr>
<td>headerHeight</td>
<td>N</td>
<td>Format: length; height of the header. The default value is 28.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>panelBorderStyle</td>
<td>N</td>
<td>Border style for the bottom two corners of the container. The top two corners are always round. Possible values are default, which configures the container to have square corners, and roundCorners, which defines rounded corners. To configure the top corners to be square, set cornerRadius to 0. The default value is default.</td>
</tr>
<tr>
<td>shadowDirection</td>
<td>N</td>
<td>Direction of drop shadow. Possible values are &quot;left&quot;, &quot;center&quot;, and &quot;right&quot;. The default value is &quot;center&quot;.</td>
</tr>
<tr>
<td>shadowDistance</td>
<td>N</td>
<td>Distance of drop shadow. Negative values move shadow above the panel. The default value is 2.</td>
</tr>
</tbody>
</table>
## Styles for cfformgroup with accordion type attribute

The following styles apply to the `cfformgroup` tag with `type="accordion"`:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headerHeight</td>
<td>N</td>
<td>Format: length; height of the accordion container buttons, in pixels. The default value is 22.</td>
</tr>
<tr>
<td>marginBottom</td>
<td>N</td>
<td>Format: length; number of pixels between the container’s bottom border and its content area. The default value is -1.</td>
</tr>
<tr>
<td>marginTop</td>
<td>N</td>
<td>Format: length; number of pixels between the container’s top border and its content area. The default value is -1.</td>
</tr>
<tr>
<td>openDuration</td>
<td>N</td>
<td>Format: time; duration, in milliseconds, of the transition from one child panel to another. The default value is 250.</td>
</tr>
<tr>
<td>verticalGap</td>
<td>N</td>
<td>Format: length; number of pixels between children in the vertical direction. The default value is -1.</td>
</tr>
</tbody>
</table>

#back to top
### Styles for cfformgroup with tabnavigator type attribute

The following styles apply to the `cfformgroup` tag with the `type="tabnavigator"`:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>horizontalAlign</td>
<td>N</td>
<td>Horizontal alignment of children. The default value is left. Possible values are left, center, and right. Because the preferred width of each tab in the tab navigator container is the size of the label text, use the <code>tabWidth</code> style to increase the width of the tab to a size larger than its preferred width to see different alignments.</td>
</tr>
<tr>
<td>horizontalGap</td>
<td>N</td>
<td>Format: length; number of pixels between children in the horizontal direction. The default value is 6.</td>
</tr>
<tr>
<td>tabHeight</td>
<td>N</td>
<td>Format: length; default tab height, in pixels. The default value is 22.</td>
</tr>
<tr>
<td>tabWidth</td>
<td>N</td>
<td>Format: length; width of the tabs, in pixels. If undefined, the default tab widths are automatically calculated from the label text. If the width of the container is smaller than the width of the label text, the labels are truncated. If a tab label is truncated, Flash displays a tooltip with the full label text when a user moves the mouse pointer over the tab. If you specify an explicit tab width, labels do not automatically shrink to fit if they do not fit in the available space.</td>
</tr>
</tbody>
</table>
### Styles for cfformitem with hrule or vrule type attributes

The following styles apply to the `formitem` tag with `type="hrule"` or `type="vrule"`:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Y</td>
<td>Format: color; color of the line. according to the following rules:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <code>strokeWidth</code> is 1, the color of the entire line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <code>strokeWidth</code> is 2 (default), the color of the top line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <code>strokeWidth</code> is greater than 2, the color of the top and left edges</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of the rectangle. The default value is <code>##C4CCCC</code>.</td>
</tr>
<tr>
<td>shadowColor</td>
<td>Y</td>
<td>Format: color; shadow color of the line, as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <code>strokeWidth</code> is 1, does nothing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <code>strokeWidth</code> is 2 (default), the color of the bottom line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <code>strokeWidth</code> is greater than 2, the color of the bottom and right</td>
</tr>
<tr>
<td></td>
<td></td>
<td>edges of the rectangle. The default value is <code>##D4D0C8</code>.</td>
</tr>
<tr>
<td>strokeWidth</td>
<td>Y</td>
<td>Thickness of the rule in pixels, as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <code>strokeWidth</code> is 1, the rule is a 1-pixel-wide line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <code>strokeWidth</code> is 2 (default), the rule is two adjacent 1-pixel-wide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>horizontal lines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If <code>strokeWidth</code> is greater than 2, the rule is a hollow rectangle with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-pixel-wide edges. The default value is 2.</td>
</tr>
</tbody>
</table>

[back to top](#)
### Styles for `cfinput` with radio, checkbox, button, image, or submit type attributes

The following styles apply `cfinput` tags with the following `type` attribute values:

- `button`
- `checkbox`
- `image`
- `radio`
- `submit`

In some cases, a style applies only to the subset of these input types, as specified in the description.

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>borderThickness</code></td>
<td>N</td>
<td>Thickness of border &quot;ring&quot;. A value of 0 means no border. Any value greater than 2 creates a glowing &quot;ring&quot; around the button. The default value is 3.</td>
</tr>
<tr>
<td><code>cornerRadius</code></td>
<td>N</td>
<td>Radius of corners. The default value is 5.</td>
</tr>
<tr>
<td><code>horizontalGap</code></td>
<td>N</td>
<td>Gap between the label and the image in an <code>img</code> input when <code>label Placement = &quot;left&quot;</code> or &quot;right&quot;. The default value is 2.</td>
</tr>
<tr>
<td><code>repeatDelay</code></td>
<td>N</td>
<td>Format: time; number of milliseconds to wait after the first <code>buttonDown</code> event before repeating <code>buttonDown</code> events at the <code>repeatInterval</code>. The default value is 500.</td>
</tr>
<tr>
<td><code>repeatInterval</code></td>
<td>N</td>
<td>Format: time; number of milliseconds between <code>buttonDown</code> events if you press and hold a button. The default value is 35.</td>
</tr>
<tr>
<td><code>symbolBackgroundColor</code></td>
<td>Y</td>
<td>Format: color; background color of check boxes and radio buttons. The default value is <code>##FFFFFF</code> (white).</td>
</tr>
<tr>
<td><code>symbolBackgroundDisabledColor</code></td>
<td>Y</td>
<td>Format: color; background color of check boxes and radio buttons when disabled. The default value is <code>##EFEEEEF</code> (light gray).</td>
</tr>
<tr>
<td>Property</td>
<td>Format</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>symbolBackgroundPressedColor</code></td>
<td>Y</td>
<td>Format: color; background color of check boxes and radio buttons when pressed. The default value is <code>##FFFFFF</code> (white).</td>
</tr>
<tr>
<td><code>symbolColor</code></td>
<td>Y</td>
<td>Format: color; the check mark of a check box or the dot of a radio button. The default value is <code>##000000</code> (black).</td>
</tr>
<tr>
<td><code>symbolDisabledColor</code></td>
<td>Y</td>
<td>Format: color; check mark or radio button dot color if the control is disabled. The default value is <code>##848384</code> (dark gray).</td>
</tr>
<tr>
<td><code>textRollOverColor</code></td>
<td>Y</td>
<td>Format: color; text color of the label as you move the mouse pointer over the control. The default value is <code>##2B333C</code>.</td>
</tr>
<tr>
<td><code>textSelectColor</code></td>
<td>Y</td>
<td>Format: color; text color of the label as you select the control. The default value is <code>##000000</code>.</td>
</tr>
<tr>
<td><code>verticalGap</code></td>
<td>N</td>
<td>Gap between the label and the image in an <code>&lt;img&gt;</code> input when <code>labelPlacement = &quot;top&quot;</code> or &quot;bottom&quot;. The default value is 2.</td>
</tr>
</tbody>
</table>
Styles for cftextarea tag and cfinput with text, password, or hidden type attributes

The following style applies to the following tags and tag-attribute combinations:

- textarea
- cfinput type="hidden"
- cfinput type="password"
- cfinput type="text"

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disabledColor</td>
<td>Y</td>
<td>Format: color; disabled color of the Text Area.</td>
</tr>
</tbody>
</table>
Styles for cfselect with size attribute value of 1

The following styles apply to the `<cfselect>` tag when the `size` attribute is 1; that is, if the control displays one option at a time, with a drop-down list (also known as a combobox):

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alternatingRowColors</td>
<td>Y</td>
<td>Format: comma delimited list of colors for rows in an alternating pattern. Value can be a list of two or more colors. Use only if you do not specify a backgroundColor style.</td>
</tr>
<tr>
<td>closeDuration</td>
<td>N</td>
<td>Time to close the drop-down list, in milliseconds. The default value is 250.</td>
</tr>
<tr>
<td>openDuration</td>
<td>N</td>
<td>Time to close the drop-down list, in milliseconds. The default value is 250.</td>
</tr>
<tr>
<td>rollOverColor</td>
<td>Y</td>
<td>Format: color; color of the background when the user rolls over an item. The default value is #0EFFD6.</td>
</tr>
<tr>
<td>selectionColor</td>
<td>Y</td>
<td>Format: color; color of the background when the user selects an item. The default value is #0DFC1.</td>
</tr>
</tbody>
</table>
Styles for cfselect with size attribute value greater than 1

The following styles apply to the `cfselect` tag when the `size` attribute is greater than 1; that is, if the control is a list box that displays two or more options at a time:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alternatingRowColors</td>
<td>Y</td>
<td>Type: comma-delimited list of colors for rows in an alternating pattern. Value can be a list of two or more colors.</td>
</tr>
<tr>
<td>marginBottom</td>
<td>N</td>
<td>Format: length; number of pixels between the bottom of the row and the bottom of the text in the row. The default value is 0.</td>
</tr>
<tr>
<td>marginTop</td>
<td>N</td>
<td>Format: length; number of pixels between the top of the row and the top of the text in the row. The default value is 0.</td>
</tr>
<tr>
<td>rollOverColor</td>
<td>Y</td>
<td>Format: color; color of the background when the user moves the mouse pointer over the link. The default value is <code>##0EFFD6</code>.</td>
</tr>
<tr>
<td>selectionColor</td>
<td>Y</td>
<td>Format: color; color of the background when the user selects the link. The default value is <code>##0DFFC1</code>.</td>
</tr>
<tr>
<td>selectionDuration</td>
<td>N</td>
<td>The duration of the selection animation, in milliseconds. The default value is 250. Set to 0 to disable animation.</td>
</tr>
<tr>
<td>textRollOverColor</td>
<td>Y</td>
<td>Format: color; text color when the user moves the mouse pointer over the selection. The default value is <code>##02B33C</code>.</td>
</tr>
<tr>
<td>textSelectedColor</td>
<td>Y</td>
<td>Format: color; text color when selected. The default value is <code>##005F33</code>.</td>
</tr>
</tbody>
</table>
## Styles for cfcalendar tag and cfinput with dateField type attribute

The following styles apply to the `cfcalendar` tag and `dateField` type of the `cfinput` tag:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headerColors</td>
<td>Y</td>
<td>Format: color; colors of the band at the top of the DateChooser control. Specify two values, separated by a comma. For a solid band, use the same color for both values. The default value is <code>##E6EEEE,##FFFF</code>.</td>
</tr>
<tr>
<td>rollOverColor</td>
<td>Y</td>
<td>Format: color; color of the background when the user moves the mouse pointer over the DateField. The default value is <code>##E3FFD6</code>.</td>
</tr>
<tr>
<td>selectionColor</td>
<td>Y</td>
<td>Format: color; color of the background when the user selects the DateField. The default value is <code>##CDFFC1</code>.</td>
</tr>
<tr>
<td>todayColor</td>
<td>Y</td>
<td>Format: color; color of today's date. The default value is <code>{#2B333C}</code>).</td>
</tr>
</tbody>
</table>

[back to top]
# Styles for the `cfgrid` tag

The following styles apply to the `cfgrid` tag:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>horizontalAlign</code></td>
<td>N</td>
<td>Horizontal alignment of children in the container. The default value is left. Possible values are left, center, and right.</td>
</tr>
<tr>
<td><code>horizontalGap</code></td>
<td>N</td>
<td>Number of pixels between children in the horizontal direction. The default value is 8.</td>
</tr>
<tr>
<td><code>marginBottom</code></td>
<td>N</td>
<td>Number of pixels between the container's bottom border and its content area. The default value is 0.</td>
</tr>
<tr>
<td><code>marginTop</code></td>
<td>N</td>
<td>Number of pixels between the container's top border and its content area. The default value is 0.</td>
</tr>
<tr>
<td><code>verticalAlign</code></td>
<td>N</td>
<td>Vertical alignment of children in the container. The default value is top. Possible values are top, middle, and bottom.</td>
</tr>
<tr>
<td><code>verticalGap</code></td>
<td>N</td>
<td>Number of pixels between children in the vertical direction. The default value is 8.</td>
</tr>
</tbody>
</table>
## Styles for the cftree tag

The following styles apply to the `cftree` tag:

<table>
<thead>
<tr>
<th>Style</th>
<th>Inh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alternatingRowColors</td>
<td>Y</td>
<td>Type: Array; colors for rows in an alternating pattern. Value can be an Array of two or more colors.</td>
</tr>
<tr>
<td>depthColors</td>
<td>Y</td>
<td>Type: Array; array of colors used in the Tree control, in descending order.</td>
</tr>
<tr>
<td>indentation</td>
<td>N</td>
<td>Indentation for each tree level, in pixels. The default value is 8.</td>
</tr>
<tr>
<td>openDuration</td>
<td>N</td>
<td>Format: time; length of an open or close transition, in milliseconds. The default value is 250.</td>
</tr>
<tr>
<td>rollOverColor</td>
<td>Y</td>
<td>Format: color; color of the background when the user moves the mouse pointer over the link. The default value is ##E3FFD6.</td>
</tr>
<tr>
<td>selectionColor</td>
<td>Y</td>
<td>Format: color; color of the background when the user selects the link. The default value is ##CFC1.</td>
</tr>
<tr>
<td>selectionDuration</td>
<td>N</td>
<td>The duration of the selection animation, in milliseconds. The default value is 250. Set to 0 to disable animation.</td>
</tr>
<tr>
<td>textRollOverColor</td>
<td>Y</td>
<td>Format: color; color of the text when the user moves the mouse pointer over the entry. The default value is ##02B33C.</td>
</tr>
<tr>
<td>textSelectedColor</td>
<td>Y</td>
<td>Format: color; color of the text when the user selects the entry. The default value is ##005F33.</td>
</tr>
</tbody>
</table>
Application.CFC Reference

You implement methods in Application.cfc to handle ColdFusion application events and set variables in the CFC to configure application characteristics.

- Application variables
- Method summary
- onAbort
- onApplicationEnd
- onApplicationStart
- onCFCRequest
- onError
- onMissingTemplate
- onRequest
- onRequestEnd
- onRequestStart
- onServerStart
- onSessionEnd
- onSessionStart

#back to top
Application variables

The This scope for the Application.cfc contains several built-in variables, which correspond to the attributes that you set in the `cfapplication` tag. You set the values of these variables in the CFC initialization code, before you define the CFC methods. You can access the variables in any method.

⚠️ Note

Although Windows is case-insensitive, you must always start the Application.cfc filename with an uppercase A. Both application.cfc and Application.cfc are reserved words.

⚠️ Note

If your application has an Application.cfc, and an Application.cfm or onRequestend.cfm page, ColdFusion ignores the CFM pages.

The following table briefly describes the variables that you can set to control the application behavior. For more details, see the `cfapplication` tag.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no name</td>
<td>The application name. If you do not set this variable, or set it to the empty string, your CFC applies to the unnamed application scope, which is the ColdFusion J2EE servlet context. For more information on unnamed scopes see Integrating JSP and servlets in a ColdFusion application in Interoperating with JSP pages and servlets in the Developing ColdFusion Applications.</td>
</tr>
<tr>
<td>applicationTimeout</td>
<td>Administrator value</td>
<td>Life span, as a real number of days, of the application, including all Application scope variables. Use the CFML <code>CreateTimeSpan</code> function to generate this variable's value.</td>
</tr>
<tr>
<td>authcookie.disableupdate</td>
<td>False</td>
<td>Disable update of <code>cfauthorization</code> cookie using <code>cfcookie</code> or <code>cfheader</code> tag</td>
</tr>
<tr>
<td>authcookie.timeout</td>
<td>-1</td>
<td>Auth Cookie age in days.</td>
</tr>
<tr>
<td>chartStyleDirectory</td>
<td></td>
<td>Application specific client side charting styles directory.</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>clientManagement</td>
<td>no</td>
<td>yes: enables client variables. no</td>
</tr>
<tr>
<td>clientStorage</td>
<td>Administrator value</td>
<td>Where Client variables are stored; can be cookie, registry, or the name of a data source.</td>
</tr>
<tr>
<td>customtagpaths</td>
<td>Administrator value</td>
<td>Contains ColdFusion custom tag paths. It is a comma delimited list with absolute path. To use this variable, select the Enable Per App Settings option in the Administrator Server &gt; Settings page. The settings that you define here take precedence over the custom tag paths defined in the Administrator Server Settings &gt; Mappings page for the current application.</td>
</tr>
<tr>
<td>googleMapKey</td>
<td></td>
<td>The Google Maps API key required to embed Google Maps in your web pages.</td>
</tr>
<tr>
<td>datasource</td>
<td></td>
<td>Name of the data source from which the query retrieves data.</td>
</tr>
<tr>
<td>loginStorage</td>
<td>cookie</td>
<td>Whether to store login information in the Cookie scope or the Session scope.</td>
</tr>
<tr>
<td>mappings</td>
<td>Administrator value</td>
<td>A structure that contains ColdFusion mappings. Each element in the structure consists of a key and a value. The logical path is the key and the absolute path is the value. To use this variable, select the Enable Per App Settings option in the Administrator Server Settings &gt; Settings page. The mappings that you define here take precedence over the mappings defined in the Administrator Server Settings &gt; Mappings page for the current application.</td>
</tr>
<tr>
<td>Configuration Key</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>restSettings.cfclocation</td>
<td>To publish the CFCs only in a particular location, provide comma-separated list of directories where the REST CFCs are located. The directory paths can be absolute or relative. If not set, all the CFCs from the application root are published.</td>
<td></td>
</tr>
<tr>
<td>restSettings.skipCFCWithError</td>
<td>When an error occurs, continue publishing, ignoring the CFC that has caused the exception. If true, the CFC with error is ignored and the rest of the CFCs are published. By default it is false. If set to false, in case of an error, the application itself is not published. But other registered application are published. If an error occurs during application startup, the error is printed in console. Each application has separate log files for logging the issues.</td>
<td></td>
</tr>
<tr>
<td>sessioncookie.httponly</td>
<td>Specify whether session cookies have to be set as httponly or not. i.e. accessible only to Http requests</td>
<td></td>
</tr>
<tr>
<td>sessioncookie.secure</td>
<td>Specify whether session cookies have to be set as secure or not. i.e. returned on any type of connection or only secured (https) connections</td>
<td></td>
</tr>
<tr>
<td>sessioncookie.domain</td>
<td>Domain for which the cookie should be set. This should match exactly with the domain, with which application would be accessed</td>
<td></td>
</tr>
<tr>
<td>sessioncookie.timeout</td>
<td>Session Cookie age in days</td>
<td></td>
</tr>
<tr>
<td>sessioncookie.disableupdate</td>
<td>Disable update of <code>cfid</code> and <code>cf tok en cookie</code> using <code>cfc cookie</code> or <code>cfhe ader tag</code></td>
<td></td>
</tr>
<tr>
<td>serverSideFormValidation</td>
<td>Whether to enable validation on <code>cform</code> fields when the form is submitted.</td>
<td></td>
</tr>
<tr>
<td>sessionManagement</td>
<td>Whether the application supports Session scope variables.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sessionTimeout</td>
<td>Administrator value</td>
<td>Life span, as a real number of days, of the user session, including all Session variables. Use the CFML createTimeSpan function to generate this variable's value.</td>
</tr>
<tr>
<td>setClientCookies</td>
<td>True</td>
<td>Whether to send CFID and CFTOKEN cookies to the client browser.</td>
</tr>
<tr>
<td>setDomainCookies</td>
<td>False</td>
<td>Whether to set CFID and CFTOKEN cookies for a domain (not just a host).</td>
</tr>
<tr>
<td>scriptProtect</td>
<td>Administrator value</td>
<td>Whether to protect variables from cross-site scripting attacks.</td>
</tr>
<tr>
<td>security.antisamypolicy</td>
<td></td>
<td>Specify the location of antisamy file to be used when no antisamy policy file is passed to the getSafeHTML or isSafeHTML functions. The policy file can be relative to the Application CFC path or an absolute path can be provided.</td>
</tr>
<tr>
<td>compileextforinclude</td>
<td></td>
<td>Specify the list of allowed file extensions as a comma-separated list for the cfinclude tag. Specifying a wildcard * in the list makes any file included using the cfinclude tag to be compiled. If any file included using the cfinclude tag is not found in this list, their content will be statically included. By default, files with the cfm and cfml extensions are always compiled irrespective of this setting.</td>
</tr>
<tr>
<td>Setting</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>strictnumbervalidation</td>
<td>True/False. Default is true. The <code>isValid</code> function for the integer and numeric types allowed the currency symbols at the start and commas inside the number.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>In ColdFusion 11, the <code>isValid</code> function behaves in a different way. Setting <code>strictnumbervalidation</code> to false makes the <code>isValid</code> function to behave in a way just like in the previous versions (ColdFusion 10 or earlier). Note that this setting also changes the behavior of the following tags:</td>
</tr>
</tbody>
</table>
|                    |               | - `cfargument`  
|                    |               | - `cfparam`  
<p>|                    |               | - <code>cform</code>  |
| secureJSON          | Administrator value | A Boolean value that specifies whether to add a security prefix in front of the value that a ColdFusion function returns in JSON-format in response to a remote call. The default value is the value of the Prefix serialized JSON setting in the Administrator Server Settings &gt; Settings page (which defaults to <code>false</code>). You can override this value in the <code>cffunction</code> tag. For more information see <code>Improving security in Ajax programming rules</code> and <code>techniques</code> in the <code>Developing ColdFusion Applications</code>. |
| secureJSONPrefix    | Administrator value | The security prefix to put in front of the value that a ColdFusion function returns in JSON-format in response to a remote call if the <code>secureJSON</code> setting is <code>true</code>. The default value is the value of the Prefix serialized JSON setting in the Administrator Server Settings &gt; Settings page (which defaults to <code>//</code>, the JavaScript comment character). For more information see <code>Improving security in Ajax programming rules and techniques</code> in the <code>Developing ColdFusion Applications</code>. |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serialization.preservecaseforstructkeys</td>
<td>False</td>
<td>Boolean that determines if case for struct keys should be preserved when serializing a struct to JSON.</td>
</tr>
<tr>
<td>serialization.serializequeryas</td>
<td>row</td>
<td>Determines how queries should be serialized to JSON. Possible values are row, column, and struct.</td>
</tr>
</tbody>
</table>
| welcomeFileList              |       | A comma-delimited list of names of files. Tells ColdFusion not to call the onMissingTemplate method if the files are not found. Use this variable to prevent ColdFusion from invoking the onMissingTemplate handler if all of the following items are true:  
- Your web server (for example, web.xml file) has a welcome file list with CFML pages such as index.cfm that it tries to run if a URL specifies a path ending in a directory.  
- The web server sends a request for CFML pages the welcome list to ColdFusion without first determining if the page exists.  
- You want to support directory browsing in directories that do not have any of the files on the welcome file list. You specify this variable only if the Application.cfc file also specifies an onMissingTemplate handler. It must have the same list of files as your web.xml welcome file list.  
**Note:** You do not need to use the welcomeFileList variable with most "pure" web servers, such as Apache. The welcomeFileList variable has to be used with most integrated web and application servers. |
<p>| smtpServersettings           |       | A struct that contains the following values: server, username, and password. If no value is specified, takes the value in the administrator. |</p>
<table>
<thead>
<tr>
<th>Config Parameter</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sameformfieldsasarray</td>
<td>false</td>
<td>If the form fields have the same name, ColdFusion converts the form fields as an array instead of a list. To do this, in the Application.cfc, specify the following: <code>this.sameformfieldsasarray = &quot;true&quot;</code>. Note: The empty string values will be preserved only if this is set to true.</td>
</tr>
<tr>
<td>timeout</td>
<td></td>
<td>The lifespan. Timeout set using <code>&lt;cf setting requesttimeout=&quot;&quot;/&gt;</code> overrides the timeout in the Application.cfc using <code>this.timeout=&quot;&quot;</code>.</td>
</tr>
<tr>
<td>debuggingIPAddresses</td>
<td></td>
<td>A list of IP addresses that need debugging.</td>
</tr>
<tr>
<td>enablerobustexception</td>
<td></td>
<td>Overrides the default administrator settings. It does not report compile-time exceptions.</td>
</tr>
<tr>
<td>javaSettings</td>
<td></td>
<td>A structure allowing you to specify Java class paths to be made available to your code. Valid keys are: <code>loadPaths</code> (an array of paths to include when searching for Java libraries), <code>loadColdFusionClassPath</code> (a boolean indicating if the default class path should be used, defaults to false), and <code>reloadOnChange</code> (a boolean indicating if the classpaths should be reloaded when they change, the default is false).</td>
</tr>
</tbody>
</table>

**Form fields with same name**

Assume that the form fields have same name. In this case, ColdFusion converts the form fields as an array instead of a list. To do this, in the Application.cfc, specify the following: `this.sameformfieldsasarray = "true"`. The default value is false.

**Enhancements made in ColdFusion 11**

In ColdFusion 11, you can register application-specific datasources in Application.cfc. These datasources will be specific to that application and will not be available through the Administrator. If there is a name clash with a server-wide datasource, the one specific to the application will be given the priority.
With a custom driver:

```
this.datasources.dsn3 = { "driver" = "other", "url" = "jdbc:sqlserver://localhost\MSSQL2008;databaseName=pubs;sendStringParametersAsUnicode=false;querytimeout=0;MaxPooledStatements=1000", "username" = "sa", "password" = "S33NGEv!l", "class" = "com.microsoft.sqlserver.jdbc.SQLServerDriver"};
```

The following drivers are supported:

- MSSQLServer
- Oracle
- Sybase
- DB2
- Informix
- MySQL DD
- PostgreSQL
- MSAccess
- Apache Derby Embedded
- Apache Derby Client
- MySQL5
- ODBCsocket
- Other (for custom driver)

**Enhancements made in ColdFusion 9.0.1**

Application.cfc lets you specify data source authentication details for the data source. The data source settings can now be a string or a struct. When a string, it is considered to be the data source name and authentication information is taken from the data source defined in the ColdFusion Administrator.

You can specify the authentication information using a struct value for a data source. The following are the key names:

- name: data source name
- username: Username for the data source
- password: Password for the data source

**Example**
<this.datasource={name='cfartgallery', username="user", password="passwd"}>

or

<this.datasource="cfartgallery">

⚠️ **Note**

The same convention is used for ORM default data source where you can specify the data source authentication information in the ormsettings.

The following application-specific attributes have been added for Amazon S3 integration:

- **accessKeyId**: ID for Amazon S3 account.
- **awsSecretKey**: Secret key for S3 account.
- **defaultLocation**: The default location of Amazon S3 bucket creation. A bucket on S3 storage can be in one of the following regions: {{US, EU, or US-WEST}}. The defaultLocation provided in the Application.cfc defines the default location for the bucket that you create. The default value is US.

**Example**

```cfscript
this.s3.accessKeyId = "key_ID";
this.s3.awsSecretKey = "secret_key";
this.s3.defaultLocation="location";
</cfscript>

**Application-specific In-memory file system**

You can use in-memory file system specific to applications. This enables application isolation for your virtual file system. That is, the file created in the in-memory file system by one application will not be accessible to another application.

The settings can be specified in the Application.cfc as follows:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this.inmemoryfilesystem.enabled</td>
<td>Set the value to true to enable in-memory file system for application. This is the default setting.</td>
</tr>
<tr>
<td>this.inmemoryfilesystem.size</td>
<td>Specify the memory limit in MB for the in-memory file system. You can also specify the value in the ColdFusion Administrator (Server Settings &gt; Settings &gt; Memory Limit per Application for In-Memory Virtual File System). The lesser value is considered.</td>
</tr>
</tbody>
</table>
# Method summary

The following table briefly describes the application event methods that you can implement in Application.CFC:

<table>
<thead>
<tr>
<th>Method name</th>
<th>Method runs when</th>
</tr>
</thead>
<tbody>
<tr>
<td>onAbort</td>
<td>Runs when you execute the tag <code>cfabort</code></td>
</tr>
<tr>
<td>onApplicationEnd</td>
<td>The application ends: the application times out, or the server is stopped</td>
</tr>
<tr>
<td>onApplicationStart</td>
<td>The application first starts: the first request for a page is processed or the first CFC method is invoked by an event gateway instance, or a web services or Flash Remoting CFC.</td>
</tr>
<tr>
<td>onCFCRequest</td>
<td>HTTP or AMF calls are made to an application.</td>
</tr>
<tr>
<td>onError</td>
<td>An exception that is not caught by a try/catch block occurs.</td>
</tr>
<tr>
<td>onMissingTemplate</td>
<td>ColdFusion received a request for a non-existent page.</td>
</tr>
<tr>
<td>onRequest</td>
<td>The <code>onRequestStart</code> method finishes. (This method can filter request contents.)</td>
</tr>
<tr>
<td>onRequestEnd</td>
<td>All pages in the request have been processed:</td>
</tr>
<tr>
<td>onRequestStart</td>
<td>A request starts</td>
</tr>
<tr>
<td>onSessionEnd</td>
<td>A session ends</td>
</tr>
<tr>
<td>onSessionStart</td>
<td>A session starts</td>
</tr>
<tr>
<td>onServerStart</td>
<td>A ColdFusion server starts</td>
</tr>
</tbody>
</table>

All parameters to these methods are positional. You can use any names for these parameters. When a request executes, ColdFusion runs the CFC methods in the following order:

1. `onApplicationStart` (if not run before for this application)
2. `onSessionStart` (if not run before for this session)
3. `onRequestStart`
4. `onRequest/onCFCRequest`
5. `onRequestEnd`

The `onApplicationEnd`, `onSessionEnd`, and `onError` CFCs are triggered by specific events.
onAbort

Description

Runs when you execute the tag `cfabort`.

⚠️ **Note**

If `showError` attribute is specified in `cfabort`, `onError` method is executed instead of `OnAbort`.

⚠️ **Note**

When using `cfabort`, `cflocation`, or `cfcontent` tags, the `OnAbort` method is invoked instead on `OnRequestEnd`.

Returns

Nothing

Syntax

```cfml
<cffunction name="onAbort" returnType="boolean">
  <cfargument type="string" name="targetPage" required=true/>
  ...
  <cfreturn BooleanValue /> 
</cffunction>
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>targetPage</code></td>
<td>The path from the web root to the requested CFML page.</td>
</tr>
</tbody>
</table>

Example

Application.cfc
<cfcomponent>
<cffunction name="onAbort" access="public" returntype="void" hint="Hanlde Aborted request">
  <cfargument type="String" name="targetPage" required=true/>
  <cfoutput>Target Page: #targetPage#</cfoutput>
  <!--- do stuff --->
</cffunction>
</cfcomponent>

Test.cfm

<cfabort>

#back to top
onApplicationEnd

Description

Runs when an application times out or the server is shutting down.

Syntax

```<cffunction name="onApplicationEnd" returnType="void">
  <cfargument name="ApplicationScope" required=true/>
  ...
</cffunction>```

See also

onApplicationStart, Method summary, Managing the application with Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Parameters

ColdFusion passes the following parameters to the method:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationScope</td>
<td>The application scope.</td>
</tr>
</tbody>
</table>

Returns

This method does not return a value; do not use the cfreturn tag.

Usage

Use this method for any clean-up activities that your application requires when it shuts down, such as saving data in memory to a database, or to log the application end to a file. You cannot use this method to display data on a user page, because it is not associated with a request. The application ends, even if this method throws an exception. If you call this method explicitly, ColdFusion does not end the application; it does execute the method code, but does not lock the Application scope while the method executes. Use the ApplicationScope parameter to access the application scope; you cannot reference the scope directly; for example, use Arguments.ApplicationScope.myVariable, not Application.myVariable. This method can access the Server scope directly, but it does not have access to Session or Request scopes.

⚠️ Note

The application times out only if it is inactive for the time-out period. Sessions do not end, and the onSessionEnd method is not called when an application ends. For more information, see onSessionEnd.

Example
<cffunction name="onApplicationEnd">
 <cfargument name="ApplicationScope" required=true/>
 <cflog file="#This.Name#" type="Information"
     text="Application #Arguments.ApplicationScope.applicationname# Ended" >
</cffunction>
onApplicationStart

Description

Runs when ColdFusion receives the first request for a page in the application.

Syntax

```html
<cffunction name="onApplicationStart" returnType="boolean">
  ...
  <cfreturn Boolean>
</cffunction>
```

See also

onApplicationEnd, Method summary, Managing the application with Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Returns

A Boolean value: True if the application startup code ran successfully; False, otherwise. You do not need to explicitly return a True value if you omit the cffunction tag returnType attribute.

Usage

Use this method for application initialization code; for example, use it to set Application scope variables, to determine whether a required data source or other resource is available, or to log the application start. You do not have to lock the Application scope if you set Application variables in this method, and you can reference Application scope variables as you normally do; for example, as Application.myVariable. This method can access the requested page's Variables scope only if the Application.cfc file includes an onRequest method that calls the page. If you call this method explicitly, ColdFusion does not start the application; it does execute the method code, but does not lock the Application scope while the method executes. If this method throws an uncaught exception or returns False, the application does not start and ColdFusion does not process any pages in the application. In this case, ColdFusion runs the onApplicationStart method the next time a user requests a page in the application.

Example

The following example tests for the availability of a database. If the database is not available it reports and logs the error, and does not start the application; if it is available, the method initializes two Application scope variables.
<cffunction name="onApplicationStart">
<cftry>
<!--- Test whether the DB is accessible by selecting some data. --->
<cfquery name="testDB" dataSource="cfdocexamples" maxrows="2">
    SELECT Emp_ID FROM employee
</cfquery>
<!--- If you get a database error, report an error to the user, log the
    error information, and do not start the application. --->
<cfcatch type="database">
    <cfoutput>
        This application encountered an error<br>
        Please contact support.
    </cfoutput>
    <cflog file="#This.Name#" type="error"
        text="cfdocexamples DB not available. message: #cfcatch.message#
        Detail: #cfcatch.detail# Native Error: #cfcatch.NativeErrorCode#" >
    <cfreturn False>
</cfcatch>
</cftry>
<cfscript>
    Application.availableResources=0;
    Application.counter1=1;
</cfscript>
<cfreturn True>
</cffunction>
onCFCRequest

Description

Intercepts any HTTP or AMF calls to an application based on CFC request.

Syntax

```xml
<cffunction name="oncfcRequest" returnType="void">
<cfargument type="string" name="cfcname">
<cfargument type="string" name="method">
<cfargument type="struct" name="args">
</cffunction>
```

See also

- Method summary, Handling errors in Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Parameters

ColdFusion passes the following parameters to the method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfcname</td>
<td>Fully qualified dotted path to the CFC.</td>
</tr>
<tr>
<td>method</td>
<td>The name of the method invoked.</td>
</tr>
<tr>
<td>args</td>
<td>The arguments (struct) with which the method is invoked.</td>
</tr>
</tbody>
</table>

Usage

Whereas onRequest handles only requests made to ColdFusion templates, this function controls Ajax, Web Service, and Flash Remoting requests.

Example

Create a folder onCFCRequest in your web root. Place test.cfc and Application.cfm in this directory and make an HTTP call to the CFC using the following URL: [http://localhost:8500/onCFCRequest/test.cfc?method=foo&arg1=1&arg2=2&arg3=3](http://localhost:8500/onCFCRequest/test.cfc?method=foo&arg1=1&arg2=2&arg3=3) When you run the URL, the method onCFCRequest is called and the function name foo is passed along with the arguments arg1, arg2, and arg3. You can then invoke the test.cfc as shown in the following example:
<!--- Application.cfc --->
<cfcomponent>
  <cfset this.name = "onCFCRequest">
  <cffunction name="onCFCRequest">
    <cfargument type="string" name="cfcname" required=true>
    <cfargument type="string" name="method" required=true>
    <cfargument type="struct" name="args" required=true>
      <cflog text="onCFCRequest()">
        <cfdump var="#arguments#" output="console" format="text">
        <cfinvoke
          component = "arguments.cfcname"
          method = "arguments.method"
          returnVariable = "result"
          argumentCollection = "#arguments.args#">
          <cfdump var="#result#" output="console" format="text">
        </cffunction>
      <cffunction name="onRequest" output="yes" access="remote">
        <cfargument type="string" name="targetpage">
        <cflog text="onRequest()">
          <cfreturn arguments>
        </cffunction>
      </cfcomponent>
    </cffunction>
  </cfcomponent>
</cfcomponent>

<!--- test.cfc --->
<cfcomponent>
  <cffunction name="foo">
    <cfargument name="arg1" type="string">
    <cfargument name="arg2" type="string">
    <cfargument name="arg3" type="string">
    <cfreturn arguments>
   </cffunction>
  </cfcomponent>
onError

Description

Runs when an uncaught exception occurs in the application.

Syntax

```
<cffunction name="onError" returnType="void">
  <cfargument name="Exception" required=true/>
  <cfargument name="EventName" type="String" required=true/>
  ...
</cffunction>
```

See also

Method summary, Handling errors in Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Parameters

ColdFusion passes the following parameters to the method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>The ColdFusion Exception object. For information on the structure of this object, see the description of the cfcatch variable in the cfcatch description.</td>
</tr>
<tr>
<td>EventName</td>
<td>The name of the event handler that generated the exception. If the error occurs during request processing and you do not implement an onRequest method, EventName is the empty string.</td>
</tr>
</tbody>
</table>

Returns

This method does not return a value; do not use the cfreturn tag.

Usage

Use this method to handle errors in an application-specific manner. This method overrides any error handlers that you set in the ColdFusion Administrator or in cferror tags. It does not override try/catch blocks. Whether the onError method can display output depends on where the error takes place, as follows:

- The onError method can display a message to the user if an error occurs during an onApplicationStart, onSessionStart, onRequestStart, onRequest, or onRequestEnd event method, or while processing a request.
- The onError method cannot display output to the user if the error occurs during an onApplicationEnd or onSessionEnd event method, because there is no available page context; however, it can log an error.
message.
If the `onError` event handler is triggered by a scope-specific event method, such as `onSessionStart`, the error prevents further processing at the level of that scope and any lower scopes. An `onError` event triggered by an `onSessionStart` method, for example, prevents further processing in the session, but not in the application. If an exception occurs while processing the `onError` method, or if the `onError` method uses a `cfthrow` tag, the ColdFusion standard error handling mechanisms handle the exception. These mechanisms include: any error handlers specified by `cferror` tags in the Application.cfc initialization code, the site-wide error handler specified in the ColdFusion Administrator, and ColdFusion default error page. Therefore, you can use the `onError` method as a filter to handle selected errors, and use other ColdFusion error-handling techniques for the remaining errors.

**Example**

```xml
<cffunction name="onError">
    <cfargument name="Exception" required=true/>
    <cfargument type="String" name="EventName" required=true/>
    <!--- Log all errors. --->
    <cflog file="#This.Name#" type="error">
        text="Event Name: #Arguments.EventName#" >
    <cflog file="#This.Name#" type="error">
        text="Message: #Arguments.Exception.message#">
    <cflog file="#This.Name#" type="error">
        text="Root Cause Message: #Arguments.Exception.rootcause.message#"/>
    <!--- Display an error message if there is a page context. --->
    <cfif NOT (Arguments.EventName IS "onSessionEnd") OR (Arguments.EventName IS "onApplicationEnd")>
        <cfoutput>
            <h2>An unexpected error occurred.</h2>
            <p>Please provide the following information to technical support:</p>
            <p>Error Event: #Arguments.EventName#</p>
            <p>Error details:<br>
                #Arguments.Exception#</p>
        </cfoutput>
    </cfif>
</cffunction>
```
onMissingTemplate

Description

Runs when a request specifies a non-existent CFML page.

Syntax

```
<cffunction name="onMissingTemplate" returnType="boolean">
  <cfargument type="string" name="targetPage" required=true/>
  ...
  <cfreturn BooleanValue />
</cffunction>
```

See also

Method summary, Handling errors in Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Parameters

ColdFusion passes the following parameters to the method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetPage</td>
<td>The path from the web root to the requested CFML page.</td>
</tr>
</tbody>
</table>

Returns

A Boolean value. True or no return value specifies that the event has been processed. False specifies that the event was not processed.

Usage

ColdFusion invokes this method when it encounters a file not found condition, that is, when a URL specifies a CFML page that does not exist. The onMissingTemplate function must return true to indicate that the event has been processed, or return false to indicate that the event has not been processed. If the function does not return a value, it is assumed to be true. If the function returns false, ColdFusion invokes the standard error handler. If an error occurs within the onMissingTemplate function, the error handler is not invoked. Therefore, you must use try/catch blocks in your missing template handler and, if the catch block cannot handle the error, it must set the function return value to false so the standard error handler can report the error. If the onMissingTemplate function is invoked, the onApplicationStart and onSessionStart event handlers are first invoked, if appropriate, but the onRequestStart, onRequest and onRequestEnd handlers are not invoked, and processing of the request terminates when the onMissingTemplate handler returns. All standard scopes, including the Application, Session, and Client scopes, are available in the onMissingTemplate function, if they are enabled. To include the contents of a page in the onMissingTemplate function, use the cfinclude tag. Do not any other method to
include or redirect other page content, including tags and functions such as `cflocation`, `getPageContext().forward()` and `getPageContext().include()`.) Use the `{{This.welcomeFileList variable to keep this function from executing if all of the following are true:

- Your web server uses a welcome file list with one or more CFML files (such as index.cfm), that it tries to access when a user enters a URL that ends with a directory name.
- The web server sends a request for a CFML page on the welcome list to ColdFusion without first determining if the page exists.
- You want to allow users to browse web directories that do not have any files on the list.

For more information, see `welcomeFileList` in Application variables.

Example

```cfc
<!--- The web.xml welcome-file-list includes index.cfm. To allow web browsing, specify index.cfm in This.welcomeFileList. --->
<cfset This.welcomeFileList="index.cfm">

<cffunction name="onMissingTemplate">
  <cfargument name="targetPage" type="string" required=true/>
  <!--- Use a try block to catch errors. --->
  <cftry>
    <!--- Log all errors. --->
    <cflog type="error" text="Missing template: #Arguments.targetPage#">
    <!--- Display an error message. --->
    <cfoutput>
      <h3>#{Arguments.targetPage} could not be found.</h3>
      <p>You requested a non-existent ColdFusion page.<br />
        Please check the URL.</p>
    </cfoutput>
  </cftry>
  <cfreturn true />
  </cfcatch>
</cffunction>
```

#back to top
onRequest

Description

Runs when a request starts, after the onRequestStart event handler. If you implement this method, it must explicitly call the requested page to process it.

Syntax

```cfc
<cffunction name="onRequest" returnType="void">
  <cfargument name="targetPage" type="String" required=true/>
  ...
  <cfinclude template="#Arguments.targetPage#">
  ...
</cffunction>
```

See also

onRequestStart, onRequestEnd, Method summary, Managing requests in Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Parameters

ColdFusion passes the following parameters to the method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetPage</td>
<td>Path from the web root to the requested page.</td>
</tr>
</tbody>
</table>

Returns

This method does not return a value; do not use the `cfreturn` tag.

Usage

This event handler provides an optional request filter mechanism for CFML page requests (that is, .cfm pages requested using a browser). Use it to intercept requests to target pages and override the default behavior of running the requested pages. The following rules specify where and how you use the `onRequest` method.

- Implement this method only if the following are true:
  - The directory, and any subdirectories affected by this Application.cfc contain CFM files. The affected directory and subdirectories do not contain any CFC files that are intended to be accessed as web services, AJAX bind, using Flash Remoting, or using an event gateway.
  - You want to intercept the request and process it in a special way.
- If you do not implement this method, ColdFusion automatically calls the target page (or the CFC for a web service, Flash Remoting, or event gateway event).
- If you implement this method, it must explicitly call the target page, normally by using a `cfinclude` tag.
- Do not implement the `onRequest` method in any Application.cfc file that affects .cfc files that implement web services, process Flash Remoting or event gateway requests; ColdFusion does not execute the requests if you implement this method.
• Code in this method that precedes the call to the target page can perform the same functions as the `onRequestStart` method, and shares the Variables scope with the target page.
• Code in this method that follows the call to the target page can perform the same functions as the `onRequestEnd` method, and shares the Variables scope with the target page.
• If you implement this method, you can also implement the `onRequestStart` and `onRequestEnd` methods. You can use this method to do preprocessing that is required for all requests. Typical uses include filtering and modifying request page contents (such as removing extraneous white space), or creating a switching mechanism that determines the exact page to display based on available parameters.

Example

```cfc
<cffunction name="onRequest">
  <cfargument name="targetPage" type="String" required=true/>
  <cfset var content="">
  <cfsavecontent variable="content">
    <cfinclude template="#Arguments.targetPage#">
  </cfsavecontent>
  <cfoutput>
    #replace(content, "report", "MyCompany Quarterly Report", "all")#
  </cfoutput>
</cffunction>
```

#back to top
onRequestEnd

Description

Runs at the end of a request, after all other CFML code.

Syntax

```
<cffunction name="onRequestEnd" returnType="void">
  <cfargument type="String" name="targetPage" required=true/>
  ...
</cffunction>
```

See also

onRequestStart, onRequest, Method summary, Managing requests in Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Parameters

ColdFusion passes the following parameters to the method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetPage</td>
<td>Path from the web root to the requested page.</td>
</tr>
</tbody>
</table>

Returns

This method does not return a value; do not use the cfreturn tag.

Usage

This method has the same purpose as the onRequestEnd.cfm page. (You cannot use an onRequestEnd.cfm page if you have an Application.cfc file for your application.) This method runs before the request terminates; therefore, it can access the page context, and can generate output. This method can be useful for gathering performance metrics, or for displaying dynamic footer information. This method can access the requested page's Variables scope only if the Application.cfc file includes an onRequest method that calls the page. You can use Request scope variables to share data with the requested page, even if the Application.cfc file does not have an onRequest method. If you call this method explicitly, ColdFusion does not end the request, but does execute the method code.

Example

The following example displays one of two footer pages depending on whether the user has logged in:The onRequestEnd method in Application.cfc contains the following code:
<cffunction name="onRequestEnd">
<cfargument type="String" name="targetPage" required=true/>
<cfset theAuthUser=getauthuser()>
<cfif theAuthUser NEQ ">
<cfinclude template="authuserfooter.cfm">
<cfelse>
<cfinclude template="noauthuserfooter.cfm">
</cfif>
</cffunction>

A simple authuserfooter.cfm page consists of the following code:

<cfoutput>
<h3>Thank you for shopping at our store, #theAuthUser#!</h3>
</cfoutput>

A simple noauthuserfooter.cfm page consists of the following code:

<cfoutput>
<h3>Remember, only registered users get all our benefits!</h3>
</cfoutput>

To test this example, implement code for logging in a user, or try the example with and without the following line in the onRequestStart Application.cfc method:

<cfloginuser name="Robert Smith" password="secret" roles="customer">

#back to top
onRequestStart

Description

Runs when a request starts.

Syntax

```html
<cffunction name="onRequestStart" returnType="boolean">
  <cfargument type="String" name="targetPage" required=true/>
  ...
  <cfreturn Boolean>
</cffunction>
```

See also

onRequest, onRequestEnd, Method summary, Managing requests in Application.cfc in Defining the application and its event handlers in Application.cfc in Developing ColdFusion Applications

Parameters

ColdFusion passes the following parameters to the method:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetPage</td>
<td>Path from the web root to the requested page.</td>
</tr>
</tbody>
</table>

Returns

A Boolean value. Return False to prevent ColdFusion from processing the request. You do not need to explicitly return a True value if you omit the cffunction tag returnType attribute.

Usage

This method runs at the beginning of the request. It is useful for user authorization (login handling), and for request-specific variable initialization, such as gathering performance statistics. If this method throws an exception (for example, if it uses the cfthrow tag), ColdFusion handles the error and does not process the request further. If you call this method explicitly, ColdFusion does not start a request, but does execute the method code. This method can access the requested page’s Variables scope only if the Application.cfc file includes an onRequest method that calls the page. You can use Request scope variables to share data with the requested page even if Application.cfc does not have an onRequest method.

Example

This example uses the authentication code generated by the ColdFusion Dreamweaver Login wizard to ensure that the user is logged in. The wizard generates code that is appropriate for Application.cfm only. To use this code with the Application.CFC, delete the generated Application.CFM
<cffunction name="onRequestStart">
  <cfargument name="requestname" required=true/>

  <!--- Authentication code, generated by the Dreamweaver Login wizard. --->
</cffunction>

<cfinclude template="mm_wizard_application_include.cfm">

<!--- Regular maintenance is done late at night. During those hours, tell people to come back later, and do not process the request further. --->
<cfscript>
  if ((Hour(now()) gt 1) and (Hour(now()) lt 3)) {
    WriteOutput("The system is undergoing periodic maintenance. Please return after 3:00 AM Eastern time.");
    return false;
  } else {
    this.start=now();
    return true;
  }
</cfscript>
</cffunction>
onServerStart

Note

Despite being documented in this section of the manual, this onServerStart method is NOT a method of Application.cfc. See below for more details on where, why, and how to use this method.

ColdFusion now supports a CFC with an onServerStart method that runs only when the server starts. The onServerStart method takes no parameters, and is the only function in the CFC. The function is useful for application-independent tasks, such as instantiating the applications, configuring logging, or setting up the scheduler.

By default, ColdFusion looks for the onServerStart method in cf_webroot/Server.cfc. To specify a different filepath:

1. Launch ColdFusion Administrator.
2. Click ColdFusion Administrator Server Settings > Settings.
3. Specify the absolute filepath under the web root on the Settings page such as c:\Server.cfc. Alternatively, you can use a dot-delimited path under the web root, such as a.b.Server.

Note

If you use an absolute path, the filename must end with .cfc. If you use a relative path or dotted path, do not end the name with the .cfc suffix.

You select an option on the Settings page to enable and disable the onServerStart method. By default, the method is disabled.

You can also specify a timeout limit (in seconds) for the onServerStart method. The timeout limit determines the duration for which the method would be allowed to run during server start up. This setting can be specified in server.cfc.

The onServerStart method can use most CFML features, but not any features that require full server start. For example, the method cannot use a cfhttp tag with a URL that specifies a location on the same server.

You also cannot use Application or Request scope variables in the method.

By default, all errors, including any serverCFC errors, are logged in <ColdFusion_home>/WEB-INF/cfusion/logs directory for standalone and <appserver_root>/logs directory for J2EE configurations.

You can also specify a different location for logging by configuring the log directory setting in ColdFusion Administrator > Debugging and Logging > Logging Settings.

The server.log file contains server startup information. So, any server.CFC startup errors are logged in it, but for details about the error, you have to see the exception.log file. In addition, server startup information is logged in {appserver_root}/logs directory.

For WebSphere, it is logged in the SystemOut.log file.
onSessionEnd

Description

Runs when a session ends.

Syntax

```cfc
<cffunction name="onSessionEnd" returnType="void">
  <cfargument name="SessionScope" required=True/>
  <cfargument name="ApplicationScope" required=False/>
  ...
</cffunction>
```

See also

See also onSessionStart, Method summary, Managing requests in Application.cfc in Defining the application and its event handlers in Application.cfc in the Developing ColdFusion Applications

Parameters

ColdFusion passes the following parameters to the method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SessionScope</td>
<td>The Session scope</td>
</tr>
<tr>
<td>ApplicationScope</td>
<td>The Application scope</td>
</tr>
</tbody>
</table>

Returns

This method does not return a value; do not use the cfreturn tag.

Usage

Use this method for any clean-up activities when the session ends. A session ends when the session is inactive for the session time-out period. You can, for example, save session-related data, such as shopping cart contents or whether the user has not completed an order, in a database, or do any other required processing based on the user's status. You might also want to log the end of the session, or other session-related information, to a file for diagnostic use. If you call this method explicitly, ColdFusion does not end the session; it does execute the method code, but does not lock the Session. You cannot use this method to display data on a user page, because it is not associated with a request. You can access shared scope variables as follows:

- Use the SessionScope parameter to access the Session scope. You cannot reference the Session scope directly; for example, use Arguments.SessionScope.myVariable, not Session.myVariable.
- You must use the ApplicationScope parameter to access the Application scope. You cannot reference the Application scope directly; for example, use Arguments.ApplicationScope.myVariable, not Application.myVariable. Use a named lock when you reference variables in the Application scope, as shown in the example.
- You can access the Server scope directly; for example, Server.myVariable.
You cannot access the Request scope. Sessions do not end, and the `onSessionEnd` method is not called when an application ends. The `onSessionEnd` does not execute if there is no active application, however.

**Example**

The following method decrements an Application scope session count variable and logs the session length.

```cfc
<cffunction name="onSessionEnd">
    <cfargument name = "SessionScope" required=true/>
    <cfargument name = "AppScope" required=true/>
    <cfset var sessionLength = TimeFormat(Now() - SessionScope.started, "H:mm:ss")>
    <cflock name="AppLock" timeout="5" type="Exclusive">
        <cfset Arguments.AppScope.sessions = Arguments.AppScope.sessions - 1>
    </cflock>
    <cflog file="#This.Name#" type="Information"
        text="Session #Arguments.SessionScope.sessionid# ended. Length: #sessionLength# Active sessions: #Arguments.AppScope.sessions#">
</cffunction>
```

#back to top
onSessionStart

Description

Runs when a session starts.

Syntax

```html
<cffunction name="onSessionStart" returnType="void">
  ...
</cffunction>
```

See also

`onSessionEnd`, `Method summary`, `Managing requests in Application.cfc` in `Defining the application and its event handlers in Application.cfc` in the `Developing ColdFusion Applications`

Returns

This method does not return a value; do not use the `cfreturn` tag.

Usage

This method is useful for initializing Session scope data, such as a shopping cart, or setting session-specific Application scope variables, such as for tracking the number of active sessions. You need not lock the Session scope to set its variables using this method. If you call this method explicitly, ColdFusion does not start a session; it does execute the method code, but does not lock the Session scope. This method can access the requested page's Variables scope only if the Application.cfc file includes an `onRequest` method that calls the page.

Example

The following `onSessionStart` example initializes some Session scope variables and increments an Application scope counter of active sessions.

```html
<cffunction name="onSessionStart">
  <cfscript>
    Session.started = now();
    Session.shoppingCart = StructNew();
    Session.shoppingCart.items = 0;
  </cfscript>
  <cflock scope="Application" timeout="5" type="Exclusive">
    <cfset Application.sessions = Application.sessions + 1>
  </cflock>
</cffunction>
```
ColdFusion Event Gateway Reference

Java interfaces are available for building ColdFusion custom CFXs in Java.

⚠️ Note

The following CFML functions also apply to gateway application development: GetGatewayHelper, SendGatewayMessage.

- addEvent
- CFEvent
- CFEventclass
- Constructor
- Gateway development interfaces and classes
  - GatewayHelper interface
  - Gateway interface
  - GatewayServices class
- getCFCMethod
- getCFPath
- setCFCTimeout
- getData
- getGatewayID
- getGatewayID_1
- getGatewayServices
- getGatewayType
- getHelper
- getLogger
- getMaxQueueSize
- getOriginatorID
- setCFCMethod
- setCFCPath
- getQueueSize
- getStatus
- outgoingMessage
- restart
- setCFCListeners
- setData
- setGatewayID
- setGatewayType
- setOriginatorID
- start
- stop
- Logger class
- debug
- error
- fatal
- info
- warn
- CFML CFEvent structure
- IM gateway methods and commands
- IM Gateway CFC incoming message methods
- `onAddBuddyRequest`
- `onAddBuddyResponse`
- `onBuddyStatus`
- `onIMServerMessage`
- `onIncomingMessage`
- **IM Gateway GatewayHelper class methods**
- **IM gateway message sending commands**
- `addBuddy`
- `addDeny`
- `addPermit`
- `getBuddyInfo`
- `getBuddyList`
- `getCustomAwayMessage`
- `getDenyList`
- `getNickName`
- `getPermitList`
- `getPermitMode`
- `getProtocolName`
- `getStatusAsString`
- `getStatusTimeStamp`
- `numberOfMessagesReceived`
- `numberOfMessagesSent`
- `removeBuddy`
- `removeDeny`
- `removePermit`
- `setNickName`
- `setPermitMode`
- `setStatus`
- **SMS Gateway CFEvent structure and commands**
- **SMS Gateway incoming message CFEvent structure**
- **SMS gateway message sending commands**
- `submit command`
- `submit Multi command`
- `data command`
- **CFML event gateway SendGatewayMessage data parameter**

#back to top
addEvent

Description

Sends a CFEvent instance to ColdFusion for dispatching to a listener CFC.

Category

Event Gateway Development

Syntax

```java
boolean addEvent(CFEvent msg)
```

See also

getMaxQueueSize, getMaxQueueSize, Responding to incoming messages in Building an event gateway in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>The CFEvent object containing the message to be queued for delivery to the listener CFC.</td>
</tr>
</tbody>
</table>

Returns

True if the event was added to the gateway services queue for delivery, false, otherwise. Therefore, a true response does not indicate that the message was delivered.

Usage

The event gateway must use this method to send incoming messages to the application for processing.

Example

The following example from the ColdFusion SocketGateway code sends an event to all listener CFCs:
for (int i = 0; i < listeners.length; i++) {
    String path = listeners[i];
    CFEvent event = new CFEvent(gatewayID);
    Hashtable mydata = new Hashtable();
    mydata.put("MESSAGE", theInput);
    event.setData(mydata);
    event.setGatewayType("SocketGateway");
    event.setOriginatorID(theKey);
    event.setCfcMethod(cfcEntryPoint);
    event.setCfcTimeOut(10);
    if (path != null)
        event.setCfcPath(path);
    boolean sent = gatewayService.addEvent(event);
    if (!sent)
        log.error("SocketGateway(" + gatewayID + ") Unable to put message on
                event queue. Message not sent from " + gatewayID + ", thread " +
                theKey + ".Message was " + theInput);
}
CFEvent

Description

CFEvent constructor.

Category

Event Gateway Development

Syntax

```
CFEvent(String gatewayID)
```

See also

gatewayID, CFML_CFEvent_structure, CFCevent class in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayID</td>
<td>The ID of the gateway. This parameter indicates the source of the message and must be the value that is passed in the Gateway constructor or set using the Gateway setGatewayID method. The SMS gateway ID must be 21 characters or fewer.</td>
</tr>
</tbody>
</table>

Usage

This method creates a container for an event gateway message that you send to ColdFusion gateway services in a gatewayServices.addEvent method for delivery to a CFC listener method.

Example

The following example, based on code for the ColdFusion asynchronous CFML gateway, sends a message to that the gateway has received to a CFC:
public String outgoingMessage(coldfusion.eventgateway.CFEvent cfmsg) {
    // Get the data
    Map data = cfmsg.getData();
    boolean status = true;
    if (data != null) {
        // create an event
        CFEvent event = new coldfusion.eventgateway.CFEvent(gatewayID);
        // set the event field values
        event.setGatewayType("CFMLGateway");
        event.setOriginatorID("CFMLGateway");
        event.setData(data);
        // send it to the event service
        status = gatewayService.addEvent(event);
    }
    return new Boolean(status).ToString();
}
The Gateway class sends and receives `CFEvent` instances to communicate with the ColdFusion listener CFC or application. The `CFEvent` instances correspond to the ColdFusion application listener CFC methods and receive and contain the message structures that ColdFusion application code sends to the gateway.

- The Gateway notifies ColdFusion of a message by sending a `CFEvent` instance in `GatewayServices.addEvent` method.
- The Gateway receives a `CFEvent` instance when ColdFusion calls the gateway’s `outgoingMessage` method.

The `CFEvent` Class extends the `java.util.Hashtable` class and has the following methods:

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CFEvent</code> (String gatewayID)</td>
<td><code>CFEvent</code> constructor.</td>
</tr>
<tr>
<td>String <code>getGatewayID</code>()</td>
<td>Returns the gateway ID (set in the <code>CFEvent</code> constructor).</td>
</tr>
<tr>
<td>void <code>setCFCMethod</code> (String method) String <code>getCFCMethod</code>()</td>
<td>Sets or gets the name of the CFC method that receives an incoming message.</td>
</tr>
<tr>
<td>void <code>setCFCPath</code> (String path) String <code>getCFCPath</code>()</td>
<td>Sets or gets the path to the application listener CFC that processes the event.</td>
</tr>
<tr>
<td>void <code>setCFCTimeout</code> (String seconds) String <code>getCFCTimeout</code>()</td>
<td>Sets or gets the time-out, in seconds, for the listener CFC to process the event request.</td>
</tr>
<tr>
<td>void <code>setData</code> (Map data) Map <code>getData</code>()</td>
<td>Sets or gets the event data structure, which contains the message contents and any other gateway-specific information.</td>
</tr>
<tr>
<td>void <code>setGatewayType</code> (String type) String <code>getGatewayType</code>()</td>
<td>Sets or gets the event gateway type identifier, such as SMS.</td>
</tr>
<tr>
<td>void <code>setOriginatorID</code> (String id) String <code>getOriginatorID</code>()</td>
<td>Sets or gets the gateway- or protocol-specific identity of the originator of a message.</td>
</tr>
</tbody>
</table>
Constructor

Description

Instantiates a gateway.

Category

Event Gateway Development

Syntax

```
public void gatewayName()
public void gatewayName(String id)
public void gatewayName(String id, String configFile)
```

See also

`setGatewayID`, Class constructor in Building an event gateway in the Developing ColdFusion Applications.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The identifier for the gateway instance</td>
</tr>
<tr>
<td>configFile</td>
<td>The absolute path to the gateway configuration file.</td>
</tr>
</tbody>
</table>

Usage

If your gateway requires a configuration file, use the constructor with two parameters. Otherwise, you can use either the default constructor or the single parameter version; ColdFusion always uses the `setGatewayID` method to set the ID.

Example

The following example shows the two argument constructor implemented in the ColdFusion SocketGateway class:
public SocketGateway(String id, String configpath) {
    propsFilePath=configpath;
    try {
        FileInputStream propsFile = new FileInputStream(propsFilePath);
        properties.load(propsFile);
        propsFile.close();
        this.loadProperties();
    } catch (FileNotFoundException f) {
        // do nothing. use default value for port.
    } catch (IOException e) {
        e.printStackTrace();
    }
    gatewayID = id;
    gatewayService = GatewayServices.getGatewayServices();
}
Gateway development interfaces and classes

The ColdFusion event gateway system is defined in the coldfusion.eventgateway package. Gateway developers implement two interfaces and use several classes, as follows:
GatewayHelper interface

coldfusion.eventgateway.GatewayHelper
ColdFusion includes a coldfusion.eventgateway.GatewayHelper Java marker interface, with no methods. Implement this interface to define a class that provides gateway-specific utility methods to the ColdFusion application or listener CFC. For example, an instant messaging event gateway might use a helper class to provide buddy list management methods to the application. The Gateway class must implement a `getHelper` method that returns the helper class, or null if you do not implement the interface.
For information on GatewayHelper classes, see `GatewayHelper class`. 

#back to top
Gateway interface

coldfusion.eventgateway.Gateway

Interface for implementing ColdFusion event gateways. A class that implements this interface defines a ColdFusion event gateway type that you can use in ColdFusion applications. The class must implement the following methods:

<table>
<thead>
<tr>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GatewayName([String id[, StringconfigFile]])</td>
<td>The gateway constructor.</td>
</tr>
<tr>
<td>String getGatewayID()</td>
<td>Returns the gateway ID.</td>
</tr>
<tr>
<td>GatewayHelper getHelper()</td>
<td>Returns an instance of the GatewayHelper class for this gateway type. instance, or null if the gateway does not have a GatewayHelper class.</td>
</tr>
<tr>
<td>int getStatus()</td>
<td>Gets the event gateway status.</td>
</tr>
<tr>
<td>String outgoingMessage(coldfusion.eventgateway.CFEvent cfmessage)</td>
<td>Handles a message sent by ColdFusion and processes it to send to a message receiver.</td>
</tr>
<tr>
<td>void restart()</td>
<td>Restarts a running event gateway.</td>
</tr>
<tr>
<td>void setCFCListeners(String[] listeners)</td>
<td>Identifies the CFCs that listen for incoming messages from the event gateway.</td>
</tr>
<tr>
<td>void setGatewayID(String id)</td>
<td>Sets the gateway ID that uniquely identifies the Gateway instance.</td>
</tr>
<tr>
<td>void start()</td>
<td>Starts the event gateway.</td>
</tr>
<tr>
<td>void stop()</td>
<td>Stops the event gateway.</td>
</tr>
</tbody>
</table>
GatewayServices class

coldfusion.eventgateway.GatewayServices

The Gateway class uses the coldfusion.eventgateway.GatewayServices class to interact with the ColdFusion event gateway services. This class has the following methods:

<table>
<thead>
<tr>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GatewayServices getGatewayServices()</td>
<td>Static method that returns the GatewayServices object.</td>
</tr>
<tr>
<td>boolean addEvent(CFEvent msg)</td>
<td>Sends a CFEvent instance to ColdFusion for dispatching to a listener CFC.</td>
</tr>
<tr>
<td>coldfusion.eventgateway.Logger getLogger(String logfile)</td>
<td>Returns a ColdFusion logger object that the event gateway can use to log information in a file.</td>
</tr>
<tr>
<td>int getMaxQueueSize()</td>
<td>Returns the maximum size of the ColdFusion event queue, as set in the ColdFusion Administrator.</td>
</tr>
<tr>
<td>int getMaxQueueSize()</td>
<td>Returns the current size of the ColdFusion event queue that handles all messages for all gateways.</td>
</tr>
</tbody>
</table>
getCFCMethod

Description

Gets the name of the CFC method that processes the message.

Category

Event Gateway Development

Syntax

```
String getCFCMethod()
```

See also

getCFCPath, getCFCTimeout, setCFCMethod, CFML CFEvent structure, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

Returns

For incoming messages, the name of the method that gateway services call in the listener CFC, as set by the setCFCMethod method. If setCFCMethod has not been called, returns null, and not onIncomingMessage, which ColdFusion gateway services uses by default. Outgoing messages that are returned by a CFC in response to an incoming message also have the CFC method name in this field if the gateway set the field on the incoming message.

Usage

Most event gateways do not need to use this method. This method could be useful if a gateway sends messages to multiple CFC Methods and must determine which method is responding.

#back to top
**getCFCPath**

**Description**

Gets the path to the listener CFC that processes this message.

**Category**

Event Gateway Development

**Syntax**

```java
String getCFCPath()
```

**See also**

`getCFCMethod`, `getCFCTimeout`, `setCFCPath`, `CFML CFEvent structure`, `CFEvent class` in Event gateway elements in the Developing ColdFusion Applications

**Returns**

An absolute path to the application listener CFC that processes the event, as set by the `setCFCPath` method. If the `setCFCPath` method has not been called, returns null, not the path specified in the ColdFusion Administrator and used by default by gateway services. Outgoing messages that are returned by a CFC in response to an incoming message also have the CFC method name in this field if the gateway set the field on the incoming message.

**Usage**

Most event gateways do not need to use this method. This method could be useful if a gateway sends messages to multiple CFCs and must determine which CFC is responding.----

#back to top
**setCFTimeout**

**Description**

Sets the time-out, in seconds, during which the listener CFC must process the event request and return before ColdFusion gateway services terminates the request.

**Category**

Event Gateway Development

**Syntax**

```java
void setCFTimeout(String timeout)
```

**See also**

getCFCTimeout, setCFCMethod, setCFCPath, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeout</td>
<td>The CFC time-out period, in seconds.</td>
</tr>
</tbody>
</table>

**Usage**

When ColdFusion calls a listener CFC method to process the event, and the CFC does not return in the specified time-out period, ColdFusion terminates the request and logs an error in the application.log file. If you do not use this method, ColdFusion uses the Timeout Request value set on the Server Settings page in the ColdFusion Administrator. Use this method if your messages require a longer or shorter time-out period than standard ColdFusion HTML requests.

**Example**

The following example code is based on the Socket gateway processInput method that takes input from the socket and sends it to the CFC listener methods. It sets the CFC time-out to 10 seconds.
for (int i = 0; i < listeners.length; i++)
{
    String path = listeners[i];
    CFEvent event = new CFEvent(gatewayID);
    Hashtable mydata = new Hashtable();
    mydata.put("MESSAGE", theInput);
    event.setData(mydata);
    event.setGatewayType("SocketGateway");
    event.setOriginatorID(theKey);
    event.setCfcMethod(cfcEntryPoint);
    event.setCfcTimeOut(10);
    if (path != null)
    {
        event.setCfcPath(path);
    }
    boolean sent = gatewayService.addEvent(event);
}
getCFCTimeout

Description

Gets the time-out, in seconds, for the listener CFC to process the event request.

Category

Event Gateway Development

Syntax

```java
String getCFCTimeout()
```

See also

getCFCMethod, getCFCPath, setCFTimeout, CFML CFEvent structure, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

Returns

The listener CFC time-out, in seconds, as set by the getCFCTimeout method, or null.

Usage

Most gateways do not need to use this function. When ColdFusion calls a listener CFC method to process the event, and the CFC does not process the event in the specified time-out period, ColdFusion terminates the request and logs an error in application.log file. By default ColdFusion uses the Timeout Request value set on the Server Settings page in the ColdFusion Administrator. ----
getData

Description

Returns the data Map that contains the message contents and other gateway-specific information.

Category

Event Gateway Development

Syntax

```
Map getData()
```

See also

`getData`, `CFML CFEvent structure`, `CFEvent class` in `Event gateway elements` in the `Developing ColdFusion Applications`

Returns

The event data structure, or null. This structure includes the message contents being passed by the gateway and any other gateway-specific information.

Usage

The contents of the data Map depends on the event gateway type. Typical fields include the message contents, originator ID, destination ID, and if a gateway (such as the ColdFusion SMS gateway) supports multiple commands, the command.

⚠️ Note

The returned Map object has case-insensitive keys.

Example

The following outgoingMessage method from the SocketGateway example gateway gets the message contents from the CFEvent data field of an outgoing message. If the CFEvent object does not include an OriginatorID field, it also tries to get the originator ID from the data field.
public String outgoingMessage(coldfusion.eventgateway.CFEvent cfmsg) {
    String retcode="ok";
    // Get the table of data returned from the event handler
    Map data = cfmsg.getData();
    String message = (String) data.get("MESSAGE");
    // find the right socket to write to from the socketRegistry hashtable
    if (cfmsg.getOriginatorID() != null)
        ((SocketServerThread)socketRegistry.get(cfmsg.getOriginatorID())).
            writeOutput(message);
    else if (data.get("OriginatorID") != null)
        ((SocketServerThread)socketRegistry.get(data.get("OriginatorID"))).
            writeOutput(message);
    else {
        System.out.println("cannot send outgoing message. OriginatorID is not
            available.");
        retcode="failed";
    }
    return retcode;
}
**getGatewayID**

**Description**

Returns the gateway ID that identifies the Gateway instance.

**Category**

Event Gateway Development

**Syntax**

```java
public String getGatewayID()
```

**See also**

`setGatewayID`, *Providing Gateway class service and information routines* in *Building an event gateway* in the *Developing ColdFusion Applications*.

**Usage**

This method returns a string value that is set by the `setGatewayID` method.

**Example**

The following example is the ColdFusion SocketGateway class `getGatewayID` method:

```java
public String getGatewayID()
{
    return gatewayID;
}
```
getGatewayID_1

Description

Returns the gateway ID field of the CFEvent object.

Category

Event Gateway Development

Syntax

String getGatewayID(CFEvent event)

See also

CFEvent, CFML CFEvent structure, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

Returns

The gateway ID of the CFEvent object, or null.

Usage

Most gateways do not need to use this method. The gateway ID is set in the CFEvent constructor and normally corresponds to the gateway that is handling the event.----

#back to top

© 2014 Adobe Systems Incorporated. All rights reserved.
getGatewayServices

Description

Static method that returns the GatewayServices object. Gateway code can call this method at any time, if necessary.

Category

Event Gateway Development

Syntax

```
GatewayServices getGatewayServices()
```

See also

GatewayServices class in the Developing ColdFusion Applications

Returns

The GatewayServices object.

Usage

Gateway constructors can call this method to get a convenient reference to the GatewayServices class and its methods.

Example

The following Socket gateway constructor code sets the GatewayServices variable:

```
public SocketGateway(String id)
{
    gatewayID = id;
    gatewayService = GatewayServices.getGatewayServices();
}
```

Calls to GatewayServices methods, such as the following, use the returned value.

```
boolean sent = gatewayService.addEvent(event);
```
getGatewayType

Description

Returns the gateway type field of the CFEvent object.

Category

Event Gateway Development

Syntax

```
String getGatewayType()
```

See also

getGatewayType, CFML CFEvent structure, CFEvent class in Event gateway elements in the Developing ColdFusion Applications

Returns

The gateway type of the CFEvent object, or null.

Usage

Most gateways do not need to use this method.----

#back to top
getHelper

Description
Returns an instance of the gatewayHelper class, if any for the gateway type.

Category
Event Gateway Development

Syntax

```java
public GatewayHelper getHelper()
```

See also

GatewayHelper interface: Providing Gateway class service and information routines in Building an event gateway in the Developing ColdFusion Applications.

Returns
A coldfusion.eventgateway.GatewayHelper class instance, or null if the gateway does not have a GatewayHelper class.

Usage
ColdFusion calls this method when a ColdFusion application calls the CFML GetGatewayHelper function. The application then uses the gatewayHelper object methods to call gateway-specific utility methods, such as instant message buddy management methods.

Example
The following example is the ColdFusion SocketGateway class getHelper method:

```java
public GatewayHelper getHelper()
{
    // SocketHelper class implements the GatewayHelper interface
    return new SocketHelper();
}
```

#back to top
getLogger

Description

Returns a ColdFusion Logger object that the event gateway can use to log information in a file.

Category

Event Gateway Development

Syntax

```
coldfusion.eventgateway.Logger getLogger([String logfile])
```

See also

Logger class, Logging events and using log files in Building an event gateway the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logfile</td>
<td>The name, without an extension, of a log file in the ColdFusion logs directory. ColdFusion automatically appends a .log extension to the name. If the file does not exist, ColdFusion creates it when it logs the first message. By default, ColdFusion logs to the eventgateway.log file.</td>
</tr>
</tbody>
</table>

Returns

A ColdFusion logger object

Usage

The Logger class has five methods: debug, info, warn, error, and fatal, that correspond to the severity level that is set in the log message. Each method takes a message string, a Throwable class object, or both. If you pass a Throwable object to these methods, ColdFusion writes the exception information in the exceptions.log file.

Example

The ColdFusion example DirectoryWatcherGateway includes the following line in the constructor to get a logger object:

```java
// We create our own log file, which will be named "watcher.log"
logger = gatewayService.getLogger("watcher");
```
The following code, from the start of the routine that loads information from the configuration file, uses this object to log the initialization.

```java
// Load the properties file to get our settings
protected void loadconfig() throws ServiceRuntimeException
{
    // load config
    logger.info("DirectoryWatcher (" + gatewayID + ") Initializing
               DirectoryWatcher gateway with configuration file " + config);
    .
    .
    .
```
**getMaxQueueSize**

**Description**

Returns the maximum size of the ColdFusion event queue, as set in the ColdFusion Administrator.

**Category**

Event Gateway Development

**Syntax**

```java
int getMaxQueueSize()
```

**See also**

`addEvent`, `getQueueSize`

**Returns**

The integer maximum number of messages that the gateway services queue can hold.

**Usage**

If the queue length reaches this value, the `addEvent` method does not add its message to the processing queue. You can use this method and the `getQueueSize` method to control the rate of event queuing and to help diagnose any throughput problems in your gateways.

**Example**

The following example logs the queue size, maximum queue size, and other information if a `gatewayService.addEvent` method fails to queue a message for delivery to a listener CFC. (It uses an internal method to construct the error message string.)

```java
boolean sent = gatewayService.addEvent(cfmsg);
if (!sent)
{
    logger.error(RB.getString(this, "IMGateway.cantAddToQueue",
        gatewayType, gatewayID, ((path != null) ? path : "default"),
        Integer.ToString(gatewayService.getQueueSize()),
        Integer.ToString(gatewayService.getMaxQueueSize())));
}
```

#back to top
getOriginatorID

Description

Identifies the originator of an incoming message. Some gateway types also use this field for the destination of an outgoing message.

Category

Event Gateway Development

Syntax

```java
String getOriginatorID()
```

See also

getOriginatorID, CFML CFEvent structure, CFEVENT class in Event gateway elements in the Developing ColdFusion Applications

Returns

The protocol-specific identifier of the message originator, or null.

Example

The outgoingMessage method of the SocketGateway example gateway uses the getOriginatorID method to determine the destination of an outgoing message. This way, a listener CFC that sends a response back to the originator does not have to explicitly set a destination in the return variable. If the field is empty, (as it is in messages sent by the CFML SendGatewayMessage function) the gateway tries to get the destination from the CFEVENT data field.
```java
public String outgoingMessage(coldfusion.eventgateway.CFEvent cfmsg)
{
    String retcode="ok";
    // Get the table of data returned from the event handler
    Map data = cfmsg.getData();
    String message = (String) data.get("MESSAGE");
    // find the right socket to write to from the socketRegistry hashtable
    if (cfmsg.getOriginatorID() != null)
        ((SocketServerThread)socketRegistry.get(cfmsg.getOriginatorID())).writeOutput(message);
    else if (data.get("OriginatorID") != null)
        ((SocketServerThread)socketRegistry.get(data.get("OriginatorID"))).writeOutput(message);
    else
    {
        System.out.println("cannot send outgoing message. OriginatorID is not available.");
        retcode="failed";
    }
    return retcode;
}
```
**setCFCMethod**

**Description**

Sets the name of the CFC method that processes an incoming message.

**Category**

Event Gateway Development

**Syntax**

```java
void setCFCMethod(String method)
```

**See also**

`getCFCMethod`, `setCFCPath`, `setCFCTimeout`, `CFML CFEvent structure`, `CFEvent class` in the *Developing ColdFusion Applications*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>The method in the listener CFC that ColdFusion calls to process this event. If you do not use this method in your gateway, ColdFusion invokes the <code>onIncomingMessage</code> method.</td>
</tr>
</tbody>
</table>

**Usage**

Gateways that use a single CFC listener method do not need to use this method if the listener CFC method is named `onIncomingMessage`. For the sake of consistency, Adobe recommends that any event gateway with a single listener not override this default. A gateway, such as the ColdFusion XMPP gateway, that uses different listener methods for different message types uses this method to identify the destination method.

**Example**

The following example code comes from the ColdFusion XMPP gateway incoming message handler. It creates a CFEvent object and sets the method that handles tests based on the message type.
CFEvent cfmsg = new CFEvent(gatewayID);
cfmsg.setOriginatorID(sender);
cfmsg.setGatewayType(gatewayType);
if(messageType == IMessage.IM)
{
    // default for normal messages
    cfmsg.setCfcMethod(onIncomingMessageFunction);
}
else if(messageType == IMessage.AUTH_REQUEST)
{
    cfmsg.setCfcMethod(onAddBuddyRequestFunction);
    message = "Requesting authorization to add '' + recipient + '' to '' + sender + '' buddy list and view '' + recipient + '' presence."
;
} // Code snipped here for brevity.
setCFCPath

Description

Specifies the listener CFC that processes this event.

Category

Event Gateway Development

Syntax

```java
void setCFCPath(String path)
```

See also

getcFCPath, setCFCMethod, setCFCTimeout, CFEvent class in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>An absolute path to the application listener CFC that processes the event. If you do not call this method in your gateway, ColdFusion uses the first path configured for the event gateway instance on the Event Gateways page in the ColdFusion Administrator.</td>
</tr>
</tbody>
</table>

Usage

By default, ColdFusion delivers messages to the CFC in the first path configured for the event gateway instance on the Event Gateways page in the ColdFusion Administrator. If your application supports multiple listener CFCs, use this method to set each listener CFC and then call the gatewayService.addEvent method to send the event to the CFC.

Example

The following example code is based on the Socket gateway processInput method that takes input from the socket and sends it to the CFC listener methods. The listeners variable contains an array of listener CFCS and is set by the gateways setCFCListeners method, which ColdFusion calls when it starts the gateway.
for (int i = 0; i < listeners.length; i++)
{
    String path = listeners[i];
    CFEvent event = new CFEvent(gatewayID);
    Hashtable mydata = new Hashtable();
    mydata.put("MESSAGE", theInput);
    event.setData(mydata);
    event.setGatewayType("SocketGateway");
    event.setOriginatorID(theKey);
    event.setCFCMethod(cfcEntryPoint);
    event.setCFCTimeout(10);
    if (path != null)
        event.setCFCPath(path);
    boolean sent = gatewayService.addEvent(event);
}
getQueueSize

Description

Returns the current size of the ColdFusion event queue that handles all messages for all gateways.

Category

Event Gateway Development

Syntax

```java
int getQueueSize()
```

See also

`addEvent, getMaxQueueSize`

Returns

The integer number of messages in the gateway message queue that are waiting to be delivered to CFCs.

Usage

You can use this method and the `getMaxQueueSize` method to control the rate of event queuing and to help diagnose any throughput problems in your gateways.

Example

The following example logs the queue size, maximum queue size, and other information if a `gatewayService.addEvent` method fails to queue a message for delivery to a listener CFC. (It uses an internal method to construct the error message string.)

```java
boolean sent = gatewayService.addEvent(cfmsg);
if (!sent)
{
    logger.error(RB.getString(this, "IMGateway.cantAddToQueue", 
    gatewayType, gatewayID, ((path != null) ? path : "default"), 
    Integer.ToString(gatewayService.getQueueSize()), 
    Integer.ToString(gatewayService.getMaxQueueSize())));
}
```

#back to top
getStatus

Description

Returns the gateway status.

Category

Event Gateway Development

Syntax

```java
public int getStatus()
```

See also

Providing Gateway class service and information routines in the Developing ColdFusion Applications

Returns

An integer status value. The Gateway interface defines the following status constants:

- STARTING
- RUNNING
- STOPPING
- STOPPED
- FAILED

Example

The following example is the ColdFusion SocketGateway class `getStatus` method:

```java
public int getStatus()
{
    return status;
}
```
**outgoingMessage**

**Description**

Sends a message from ColdFusion to a message receiver.

**Category**

Event Gateway Development

**Syntax**

```java
public String outgoingMessage(coldfusion.eventgateway.CFEvent message)
```

**See also**

*Responding to a ColdFusion function or listener CFC* in the *Developing ColdFusion Applications*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>A coldfusion.eventgateway.CFEvent instance containing the message to be sent.</td>
</tr>
</tbody>
</table>

**Returns**

A gateway-specific string, such as a message ID or a status indicator.

**Usage**

This method handles a message sent by ColdFusion and processes it as needed by the gateway type to send a message to the (usually external) message receiver. ColdFusion calls this method when the listener method of a listener CFC returns a message or when a ColdFusion application calls the `SendGatewayMessage` function. ColdFusion passes the String returned by this method back as the return value of a CFML `SendGatewayMessage` function.

**Example**

The following example is the ColdFusion SocketGateway class `outgoingMessage` method:
public String outgoingMessage(coldfusion.eventgateway.CFEvent cfmsg) {
    String retcode="ok";
    // Get the table of data returned from the event handler
    Map data = cfmsg.getData();
    String message = (String) data.get("MESSAGE");
    // find the right socket to write to from the socketRegistry hashtable
    if (cfmsg.getOriginatorID() != null && message != null) {
        SocketServerThread st =
            ((SocketServerThread)socketRegistry.get(cfmsg.getOriginatorID()));
        if(st != null)
            st.writeOutput(message);
        else {
            log.error("Cannot send outgoing message. OriginatorID "+
                cfmsg.getOriginatorID() + "," + " is not a valid socket id.");
            retcode="failed";
        }
    } else if (data.get("OriginatorID") != null && message != null) {
        SocketServerThread st =
            ((SocketServerThread)socketRegistry.get(data.get("OriginatorID")));
        if(st != null)
            st.writeOutput(message);
        else {
            log.error("Cannot send outgoing message. OriginatorID "+
                data.get("OriginatorID") + "," + " is not a valid socket id.");
            retcode="failed";
        }
    } else {
        log.error("Cannot send outgoing message. OriginatorID/MESSAGE is not available.");
        retcode="failed";
    }
    return retcode;
}
restart

Description
Stops a gateway if it is running and starts it up.

Category
Event Gateway Development

Syntax

```
public void restart()
```

See also

`start`, `stop`

Usage
In most cases, you implement this method as a call to the stop method followed by a start method, but you may be able to optimize the restart method based on the type of gateway.

Example
The following example is the ColdFusion SocketGateway class `restart` method:

```
public void restart()
{
    stop();
    start();
}
```
setCFCListers

Description
Sets the array of listener CFCs that the gateway sends messages to.

Category
Event Gateway Development

Syntax

```
public void setCFCListeners(String[] listeners)
```

See also
Constructor, getGatewayID, setCFCPath, Providing Gateway class service and information routines in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listeners</td>
<td>Array of absolute file paths to CFCs to which the gateway forwards messages when it gets events.</td>
</tr>
</tbody>
</table>

Usage
When ColdFusion starts a gateway instance, it calls this method with the names in the instances listener list in the ColdFusion Administrator. ColdFusion can also call this method if the ColdFusion Administrator listener list changes while the gateway is running.

Example
The following example is the ColdFusion SocketGateway class setCFCListeners method:
public void setCFCListeners(String[] listeners)
{
    ArrayList aListeners = new ArrayList();
    for(int i = 0; i<listeners.length; i++)
    {
        aListeners.add(listeners[i]);
    }
    // Try not to pull the rug out from underneath a running message
    synchronized (cfcListeners)
    {
        cfcListeners = aListeners;
    }
}
**setData**

**Description**

Adds the gateway-specific data, including any message contents, as a Java Map to the CFEvent object

**Category**

Event Gateway Development

**Syntax**

```java
void setData(Map data)
```

**See also**

`getData`, `CFML CFEvent structure`, `CFEvent class` in the *Developing ColdFusion Applications*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>The incoming message and any additional gateway-specific event data.</td>
</tr>
</tbody>
</table>

**Usage**

The number of fields and their contents depend on the event gateway type. The Map keys must be strings. Because ColdFusion is not case sensitive, it converts the Map passed in the `setData` method to a case insensitive Map. As a result, do not create entries in the data with names that differ only in case.

**Example**

The following code shows the routine from the example JMS gateway that handles incoming messages. It puts the JMS message ID and contents in a data HashMap, and uses it in the `setData` method:
public void handleMessage(String msg, String topicName, String msgID) {
    coldfusion.eventgateway.Logger log = getGatewayServices().getLogger();
    Map data = new HashMap();
    CFEvent cfMsg = new CFEvent(getGatewayID());
    data.put("msg", msg);
    data.put("id", msgID);
    cfMsg.setData(data);
    cfMsg.setOriginatorID(topicName);
    cfMsg.setGatewayType("JMS");
    if (sendMessage(cfMsg)) {
        log.info("Added message '" + msgID + '" to queue.");
    } else {
        log.error("Failed to add message '" + msgID + '" to queue.");
    }
}
**setGatewayID**

**Description**
Sets the gateway ID that uniquely identifies the Gateway instance.

**Category**
Event Gateway Development

**Syntax**

```java
public void setGatewayID(String id)
```

**See also**
Constructor, `getGatewayID`, *Providing Gateway class service and information routines* in the *Developing ColdFusion Applications*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The identifier for this gateway instance.</td>
</tr>
</tbody>
</table>

**Usage**
This method sets a string value that is returned by the `getGatewayID` method. ColdFusion calls this method to set the gateway ID with the value specified in the gateway instance configuration in the ColdFusion Administrator before it starts the event gateway, even if the Gateway constructor also sets the ID.

**Example**
The following example is the ColdFusion SocketGateway class `setGatewayID` method:

```java
public void setGatewayID(String id)
{
    gatewayID = id;
}
```

[back to top]
setGatewayType

Description

Identifies the type of event gateway.

Category

Event Gateway Development

Syntax

```java
void setGatewayType(String gatewayType)
```

See also

`getGatewayType`, `CFML CFEvent structure`, `CFEvent class` in *Developing ColdFusion Applications*

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayType</td>
<td>A gateway type identifier.</td>
</tr>
</tbody>
</table>

Usage

For the sake of consistency, use the same name in this method and in the Type Name field when you add the event gateway type in the ColdFusion Administrator. Gateway application CFCs that handle multiple gateway types, such as those in an instant messaging application that handles multiple instant messaging providers, could use this field to determine the protocol type and any gateway type-specific actions.

Example

The following code shows the routine from the example JMS gateway that handles incoming messages. It sets the gateway type to JMS:
public void handleMessage(String msg, String topicName, String msgID) {
    coldfusion.eventgateway.Logger log = getGatewayServices().getLogger();
    Map data = new HashMap();
    CFEvent cfMsg = new CFEvent(getGatewayID());
    data.put("msg", msg);
    data.put("id", msgID);
    cfMsg.setData(data);
    cfMsg.setOriginatorID(topicName);
    cfMsg.setGatewayType("JMS");
    if (sendMessage(cfMsg)) {
        log.info("Added message '" + msgID + '" to queue.");
    } else {
        log.error("Failed to add message '" + msgID + '" to queue.");
    }
}
setOriginatorID

Description

Identifies the originator of an incoming message.

Category

Event Gateway Development

Syntax

```java
void setOriginatorID(String originatorID)
```

See also

`getOriginatorID`, `CFML CFEvent structure`, `CFEvent class` in * Developing ColdFusion Applications*

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>originatorID</td>
<td>The gateway or protocol-specific ID of the message originator.</td>
</tr>
</tbody>
</table>

Example

The following code shows the routine from the example JMS gateway that handles incoming messages. It sets the originator ID to the name of the JMS topic that the gateway handles:

```java
public void handleMessage(String msg, String topicName, String msgID) {
    coldfusion.eventgateway.Logger log = getGatewayServices().getLogger();
    Map data = new HashMap();
    CFEvent cfMsg = new CFEvent(getGatewayID());
    data.put("msg", msg);
    data.put("id", msgID);
    cfMsg.setData(data);
    cfMsg.setOriginatorID(topicName);
    cfMsg.setGatewayType("JMS");
    if (sendMessage(cfMsg)) {
        log.info("Added message "+ msgID + " to queue.");
    } else {
        log.error("Failed to add message "+ msgID + " to queue.");
    }
}
```
start

Description

starts a gateway running.

Category

Event Gateway Development

Syntax

```java
public void start()
```

See also

restart, stop, Starting, stopping, and restarting the event gateway in Building an event gateway in the Developing ColdFusion Applications

Usage

Start a gateway by performing any required initialization. This method starts any listener thread or threads that monitor the gateway's event source. The ColdFusion Administrator calls this function when it starts a gateway instance. This method should update the status information that is returned by the `getStatus` method to indicate when the gateway is starting and when the gateway is running. The ColdFusion Administrator Gateway Types page lets you specify a time-out for the gateway startup, and whether to kill the gateway on startup time-out. If you enable the kill option and the `start` method does not return in the time-out period, ColdFusion kills the thread that called this function.

Example

The following example is the ColdFusion SocketGateway class `restart` method:

```java
public void start()
{
    status = STARTING;
    listening=true;
    // Start up event generator thread
    Runnable r = new Runnable()
    {
        public void run()
        {
            socketServer();
        }
    };
    Thread t = new Thread(r);
    t.start();
    status = RUNNING;
}
```
stop

Description

Stops a gateway if it is running.

Category

Event Gateway Development

Syntax

```java
public void stop()
```

See also

restart, start, Starting, stopping, and restarting the event gateway in Building an event gateway in the Developing ColdFusion Applications

Usage

Stops a gateway by performing any required clean-up operations. This method stops any listener thread or threads that monitor the gateway's event source and releases any other resources. The ColdFusion Administrator calls this function when it stops a gateway instance. This method should update the status information that is returned by the `getStatus` method to indicate when the gateway is stopping and when the gateway is stopped.

Example

The following example is the ColdFusion SocketGateway class `stop` method:
public void stop()
{
    status = STOPPING;
    listening=false;
    Enumeration e = socketRegistry.elements();
    while (e.hasMoreElements()) {
        try {
            ((SocketServerThread)e.nextElement()).socket.close();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
    if (serverSocket != null) {
        try {
            serverSocket.close();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
    status = STOPPED;
}
Logger class

coldfusion.eventgateway.Logger

⚠️ Note

This class is in the coldfusion.log package, not the coldfusion.eventgateway package, which contains all other event gateway-related interfaces and classes.

The Logger class logs messages to a file in the ColdFusion logs directory. (You set this directory on the ColdFusion Administrator Logging Settings page.) The coldfusion.eventgateway.GatewayServices.getLogger() method returns an instance of the Logger class. The Logger class has the following methods:

<table>
<thead>
<tr>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td>Writes a debugging message to the log file.</td>
</tr>
<tr>
<td>error</td>
<td>Writes an error message to the log file.</td>
</tr>
<tr>
<td>fatal</td>
<td>Writes a fatal error to the log file.</td>
</tr>
<tr>
<td>info</td>
<td>Writes an informational message to the log file.</td>
</tr>
<tr>
<td>warn</td>
<td>Writes a warning message to the log file.</td>
</tr>
</tbody>
</table>

#back to top
debug

Description

 Writes a log entry with a debugging severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

Category

Event Gateway Development

Syntax

```
depth(String message)
depth(Throwable th)
depth(String message, Throwable th)
```

See also

error, fatal, info, warn, getLogger, Logging events and using log files in Building an event gateway in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The message to include in the log entry.</td>
</tr>
<tr>
<td>th</td>
<td>A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory.</td>
</tr>
</tbody>
</table>

Usage

Use this method to send a debugging message to the ColdFusion logging subsystem. By default, ColdFusion does not write debugging messages to the log file. To have debug messages appear in the log file, change the priority entry in cf_root\lib\neo-logging.xml (in the server configuration) or cf_root\WEB-INF\cfusion\lib\neo-logging.xml (in the J2EE configuration. Change the following entry:

```
<var name='priority'>
  <string>information</string>
</var>
```

to the following:

```
<var name='priority'>
  <string>debug</string>
</var>
```
With debug priority, ColdFusion writes messages with a severity of "debug" to the log file specified in the `getLogge` method that returned the Logger instance (or the default log file).

**Example**

The ColdFusion instant messaging gateways use the following line to log information about incoming administrative messages or errors only when debugging priority is on.

```java
// code to process incoming administrative messages or errors
logger.debug(gatewayType + "Gateway (" + gatewayID + ") admin message: " + msg.getMessage());
```

#back to top
error

Description

Writes a log entry with an error severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

Category

Event Gateway Development

Syntax

```java
error(String message)
error(Throwable th)
error(String message, Throwable th)
```

See also

debug, fatal, info, warn, getLogger. Logging events and using log files in Building an event gateway in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The message to include in the log entry.</td>
</tr>
<tr>
<td>th</td>
<td>A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory.</td>
</tr>
</tbody>
</table>

Usage

Use this method to send an error message to the ColdFusion logging subsystem. ColdFusion writes messages with a severity of "error" to the log file specified in the getLogger method that returned the Logger instance (or the default log file).

Example

The ColdFusion example SocketGateway class includes the following code in the outgoingMessage method. It writes an error message if the message's originator ID does not correspond to an open socket.
SocketServerThread st =
((SocketServerThread)socketRegistry.get(cfmsg.getOriginatorID()));
if(st != null)
    st.writeOutput(message);
else {
    log.error("Cannot send outgoing message. OriginatorID "+
               cfmsg.getOriginatorID() + "' is not a valid socket id.");
    retcode="failed";
}
fatal

Description

Writes a log entry with a fatal severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

Category

Event Gateway Development

Syntax

fatal(String message)
fatal(Throwable th)
fatal(String message, Throwable th)

See also

debug, error, info, warn, getLogger, Logging events and using log files in Building an event gateway in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The message to include in the log entry.</td>
</tr>
<tr>
<td>th</td>
<td>A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory.</td>
</tr>
</tbody>
</table>

Usage

Use this method to send a fatal error message to the ColdFusion logging subsystem. ColdFusion writes message with a severity of "fatal" to the log file specified in the getLogger method that returned the Logger instance (or the default log file).

#back to top
info

Description

Writes a log entry with an information severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

Category

Event Gateway Development

Syntax

```java
info(String message)
info(Throwable th)
info(String message, Throwable th)
```

See also

decay, error, fatal, warn, getLogger. Logging events and using log files in Building an event gateway in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The message to include in the log entry.</td>
</tr>
<tr>
<td>th</td>
<td>A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory. Not normally used with this method.</td>
</tr>
</tbody>
</table>

Usage

Use this method to send an informational message to the ColdFusion logging subsystem. ColdFusion writes messages with a severity of "information" to the log file specified in the getLogger method that returned the Logger instance (or the default log file). ColdFusion normally logs all information severity messages. So do not use this severity for debugging messages or for events that happen frequently.

Example

The ColdFusion example DirectoryWatcherGateway class includes the following line at the top of its loadconfig method that loads the gateway’s configuration file. It writes a message including the gateway ID and configuration file.

```java
logger.info("DirectoryWatcher (" + gatewayID + ") Initializing DirectoryWatcher gateway with configuration file " + config);
```
**warn**

**Description**

 Writes a log entry with a warning severity to the ColdFusion logger. The entry includes the severity, thread ID, date, time, and a text message.

**Category**

 Event Gateway Development

**Syntax**

```plaintext
warn(String message)
warn(Throwable th)
warn(String message, Throwable th)
```

**See also**

`debug`, `error`, `fatal`, `info`, `getLogger`, Logging events and using log files in Building an event gateway in the Developing ColdFusion Applications

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The message to include in the log entry.</td>
</tr>
<tr>
<td>th</td>
<td>A throwable object, normally an exception. ColdFusion logs the exception information in the exception.log file in the ColdFusion logs directory.</td>
</tr>
</tbody>
</table>

**Usage**

 Use this method to send a warning message to the ColdFusion logging subsystem. ColdFusion writes messages with a severity of "warning" to the log file specified in the `getLogger` method that returned the Logger instance (or the default log file).

**Example**

 The ColdFusion example SocketWatcherGateway class includes the following code in its constructor to load a configuration file. If it cannot load the file, it converts the exception information to a string and logs a warning that includes the gateway ID, and the exception information. It also passes the exception to the `warn` method.
propsFilePath=configpath;
try {
    FileInputStream propsFile = new FileInputStream(propsFilePath);
    properties.load(propsFile);
    propsFile.close();
    this.loadProperties();
} 
catch (IOException e) {
    // do nothing. use default value for port.
    log.warn("SocketGateway(" + gatewayID + ") Unable to read configuration file 
" + propsFilePath + ": " + e.ToString() + ".Using default port.", e);
}
The CFML listener CFC methods receive messages in the form of a CFEvent structure that corresponds to the CFE\texttt{vent} class in Event gateway elements in the Developing ColdFusion Applications that gateway developers use. This structure has the following fields. Some of the fields might not be used by all gateways. All fields contain text or numeric values except the Data field, which contains a structure.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GatewayID</td>
<td>The event gateway that sent the event or will handle the outgoing message. The value is the ID of an event gateway instance configured on the ColdFusion Administrator Gateways page. If the application calls the SendGatewayMessage function to respond to the event gateway, it uses this ID as the function's first parameter.</td>
</tr>
<tr>
<td>Data</td>
<td>A structure containing the event data, including the message. The Data structure contents depend on the event gateway type. This field corresponds to the SendGatewayMessage function's second parameter.</td>
</tr>
<tr>
<td>OriginatorID</td>
<td>The originator of the message. The value depends on the protocol or event gateway type. Some event gateways might require this value in response messages to identify the destination of the response. Identifies the sender of the message.</td>
</tr>
<tr>
<td>GatewayType</td>
<td>The type of event gateway, such as SMS. An application that can process messages from multiple event gateway types can use this field. This value is the gateway type name that is specified by the event Gateway class. It is not necessarily the same as the gateway type name in the ColdFusion Administrator.</td>
</tr>
<tr>
<td>CFCPath</td>
<td>The location of the listener CFC. The listener CFC does not need to use this field.</td>
</tr>
<tr>
<td>CFCMethod</td>
<td>The listener method that ColdFusion invokes to process the event. The listener CFC does not need to use this field.</td>
</tr>
<tr>
<td>CFCTimeout</td>
<td>The time-out, in seconds, for the listener CFC to process the event request. The listener CFC does not need to use this field.</td>
</tr>
</tbody>
</table>
IM gateway methods and commands

The XMPP and IBM Sametime gateways implement CFC methods to receive messages, use the gatewayHelper object methods to manage the gateway, and use outgoing message commands to send messages.

#back to top
**IM Gateway CFC incoming message methods**

You write the following CFC methods to handle incoming messages from an XMPP or Lotus Sametime instant messaging gateway.

⚠️ **Note**

The method names assume a default gateway configuration. ColdFusion lets you change the method names and disable event types in the gateway configuration file.

<table>
<thead>
<tr>
<th>Method</th>
<th>Message type</th>
</tr>
</thead>
<tbody>
<tr>
<td>onAddBuddyRequest</td>
<td>Requests from other IM users to add the gateway ID as their buddy</td>
</tr>
<tr>
<td>onAddBuddyResponse</td>
<td>Responses from others to requests from your gateway to add them to your buddy lists. Also used by buddies to ask to be removed from your list.</td>
</tr>
<tr>
<td>onBuddyStatus</td>
<td>Online status information messages</td>
</tr>
<tr>
<td>onIMServerMessage</td>
<td>Error and administrative messages from the IM server</td>
</tr>
<tr>
<td>onIncomingMessage</td>
<td>Instant messages</td>
</tr>
</tbody>
</table>

#back to top
onAddBuddyRequest

Description

Handles incoming requests for users to add the gateway user name as one of their buddies.

Syntax

```plaintext
onAddBuddyRequest(CFEvent)
```

See also

onIncomingMessage, onAddBuddyResponse, onBuddyStatus, onIMServerMessage

Parameters

The method must take one parameter, a CFEvent structure with the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayType</td>
<td>Gateway type, either XMPP or SAMETIME</td>
</tr>
<tr>
<td>gatewayID</td>
<td>The ID of the gateway instance, as configured in ColdFusion Administrator</td>
</tr>
<tr>
<td>originatorID</td>
<td>The IM ID of the message originator</td>
</tr>
<tr>
<td>cfcMethod</td>
<td>This CFC method; by default, onAddBuddyRequest.</td>
</tr>
<tr>
<td>data.MESSAGE</td>
<td>The message that was sent with the request</td>
</tr>
<tr>
<td>data.SENDER</td>
<td>The sender's ID; identical to the originatorID field value</td>
</tr>
<tr>
<td>data.RECIPIENT</td>
<td>The recipient's ID, as specified in the gateway's configuration file</td>
</tr>
<tr>
<td>data.TIMESTAMP</td>
<td>The date and time when the message was sent</td>
</tr>
</tbody>
</table>

Returns

The function can optionally return a value to send a response message. The return structure must contain the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>command</strong></td>
<td>One of the following:</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td></td>
<td>• accept - Accept the request to add you as a buddy. ColdFusion adds the user to the permit list of users that can get status information.</td>
</tr>
<tr>
<td></td>
<td>• decline - Deny request to add you as a buddy. ColdFusion adds the user to the deny list of users that can get status information.</td>
</tr>
<tr>
<td></td>
<td>• noact - Take no action. ColdFusion does not respond to the requestor.</td>
</tr>
</tbody>
</table>

| **buddyID** | ID to which to send the message. Normally, the value of the CFEvent.data.SENDER field. Not used with the **noact** command. |

| **reason**   | A text message describing the reason for the action. Not used with the **noact** command. |

**Example**

The following example searches for the requested buddy's name in a data source and, if it finds a unique entry, adds the buddy and updates the buddy’s status information in an Application scope buddyStatus structure. If it doesn’t find the name, it declines the buddy. If there are multiple entries for the buddy name in the database, it tells the gateway not to respond. It logs all actions.
<cffunction name="onAddBuddyRequest">
  <cfargument name="CFEvent" type="struct" required="YES">
  <cfquery name="buddysearch" datasource="cfdocexamples">
    SELECT IM_ID
    FROM Employees
    WHERE IM_ID = '#CFEvent.Data.SENDER#'
  </cfquery>
  <cflock scope="APPLICATION" timeout="10" type="EXCLUSIVE">
    <cfscript>
      // If the name is in the DB once, accept; if it is missing, decline.
      // If it is in the DB multiple times, take no action.
      if (buddysearch.RecordCount IS 0) {
        action="decline";
        reason="Invalid ID";
      }
      else if (buddysearch.RecordCount IS 1) {
        action="accept";
        reason="Valid ID";
        //Add the buddy to the buddy status structure only if accepted.
        if (NOT StructKeyExists(Application, "buddyStatus")) {
          Application.buddyStatus=StructNew();
        }
        if (NOT StructKeyExists(Application.buddyStatus, CFEvent.Data.SENDER)) {
          Application.buddyStatus[#CFEvent.Data.SENDER#]=StructNew();
        }
        Application.buddyStatus[#CFEvent.Data.SENDER#].status="Accepted Buddy Request";
        Application.buddyStatus[#CFEvent.Data.SENDER#].timeStamp=CFEvent.Data.TIMESTAMP;
        Application.buddyStatus[#CFEvent.Data.SENDER#].message=CFEvent.Data.MESSAGE;
      }
      else {
        action="noact";
      }
    </cfscript>
  </cflock>
</cffunction>

<!--- Log the request and decision information. --->
<cflog file="#CFEvent.GatewayID#Status" text="onAddBuddyRequest; SENDER: #CFEvent.Data.SENDER# MESSAGE: #CFEvent.Data.MESSAGE# TIMESTAMP: #CFEvent.Data.TIMESTAMP# ACTION: #action#">
<!--- Return the action decision. --->
<cfset retValue = structNew()>
<cfset retValue.command = action>
<cfset retValue.BuddyID = CFEvent.DATA.SENDER>
<cfset retValue.Reason = reason>
<cfreturn retValue>
</cffunction>
onAddBuddyResponse

Description

Handles incoming responses from other users to requests from the gateway to be added to their buddy lists. Also receives requests from buddies to have you remove them from your buddy list.

Syntax

```onAddBuddyResponse (CFEvent)```

See also

onIncomingMessage, onAddBuddyRequest, onBuddyStatus, onIMServerMessage

Parameters

The method must take one parameter, a CFEvent structure with the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayType</td>
<td>Gateway type, either XMPP or SAMETIME.</td>
</tr>
<tr>
<td>gatewayID</td>
<td>The ID of the gateway instance, as configured in ColdFusion Administrator.</td>
</tr>
<tr>
<td>originatorID</td>
<td>The IM ID of the message originator.</td>
</tr>
<tr>
<td>cfcMethod</td>
<td>This CFC method; by default, onAddBuddyResponse.</td>
</tr>
<tr>
<td>data.MESSAGE</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• accept - The request was accepted.</td>
</tr>
<tr>
<td></td>
<td>• decline - The request was declined, or the buddy is asking you to remove them from your list.</td>
</tr>
<tr>
<td>data.SENDER</td>
<td>The sender's ID; identical to the originatorID.</td>
</tr>
<tr>
<td>data.RECIPIENT</td>
<td>The recipient's ID, as specified in the gateway's configuration file.</td>
</tr>
<tr>
<td>data.TIMESTAMP</td>
<td>The date and time when the message was sent.</td>
</tr>
</tbody>
</table>

Returns

The function does not return a value.

Example
The following example adds the buddy's status to the Application scope buddyStatus structure if the message sender accepted an add buddy request. It logs all responses.

```coldfusion
<cffunction name="onAddBuddyResponse">
<cfargument name="CFEvent" type="struct" required="YES">
<cflock scope="APPLICATION" timeout="10" type="EXCLUSIVE">
<cfscript>
//Do the following only if the buddy accepted the request.
if (NOT StructKeyExists(Application, "buddyStatus")) {
  Application.buddyStatus=StructNew();
}
if (#CFEVENT.Data.MESSAGE# IS "accept") {
  //Create a new entry in the buddyStatus record for the buddy.
  if (NOT StructKeyExists(Application.buddyStatus,
                             CFEvent.Data.SENDER)) {
    Application.buddyStatus[#CFEvent.Data.SENDER#]=StructNew();
  }
  //Set the buddy status information to indicate buddy was added.
  Application.buddyStatus[#CFEvent.Data.SENDER#].status=
    "Buddy accepted us";
  Application.buddyStatus[#CFEvent.Data.SENDER#].timeStamp=
    CFEvent.Data.TIMESTAMP;
  Application.buddyStatus[#CFEvent.Data.SENDER#].message=
    CFEvent.Data.MESSAGE;
}
</cfscript>
</cflock>
<!---- Log the information for all responses. --->
<cflog file="#CFEvent.GatewayID#Status"
  text="onAddBuddyResponse; BUDDY: #CFEvent.Data.SENDER# RESPONSE:
  #CFEvent.Data.MESSAGE# TIMESTAMP: #CFEvent.Data.TIMESTAMP#">
</cffunction>
```

#back to top
onBuddyStatus

Description
Handles incoming messages indicating online status (presence) changes of users on the gateway’s buddy list.

Syntax
onBuddyStatus(CFEvent)

See also
onIncomingMessage, onAddBuddyRequest, onAddBuddyResponse, onIMServerMessage

Parameters
The method must take one parameter, a CFEvent structure with the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayType</td>
<td>Gateway type, either XMPP or SAMETIME.</td>
</tr>
<tr>
<td>gatewayID</td>
<td>The ID of the Gateway instance, as configured in ColdFusion Administrator.</td>
</tr>
<tr>
<td>originatorID</td>
<td>The IM ID (buddy name) of the message originator.</td>
</tr>
<tr>
<td>cfcMethod</td>
<td>This CFC method; by default, onIMServerMessage.</td>
</tr>
<tr>
<td>data.BUDDYNAME</td>
<td>The sender’s buddy name, or ID; identical to the originatorID.</td>
</tr>
<tr>
<td>data.BUDDYNICKNAME</td>
<td>The buddy’s display name or nickname.</td>
</tr>
</tbody>
</table>
The buddy's status; one of the following:
- ONLINE
- OFFLINE
- AWAY
- DO NOT DISTURB
- NOT AVAILABLE
- FREE TO CHAT
- IDLE
- XMPP only
- NOT AVAILABLE
- FREE TO CHAT
- IDLE
- Sametime only
- IDLE

Use the `IMGatewayHelper.getCustomAwayMessage` method to get any custom message that the buddy sent when changing status.

The group that the buddy belongs to.

The recipient's ID, as specified in the gateway's configuration file.

The date and time when the message was sent.

⚠️ Note

You configure the buddy's nickname and group when you use the `gatewayHelper.addBuddy` method to add a buddy.

**Returns**

The function does not return a value.

**Example**

The following example keeps an Application scope structure up-to-date with a buddy's status. It also uses the `gatewayhelper.getBuddyStatus` method to get the buddy's custom away message, if any.
<cffunction name="onBuddyStatus">
<cfargument name="CFEvent" type="struct" required="YES">
<!--- Get the gatewayhelper object and to get the info for this buddy. --->
<!--- This is used to get the buddy's custom away message. --->
<cfset helper = getGatewayHelper("MYIM")>
<cfset mybuddyinfo=helper.getBuddyInfo(CFEvent.Data.BUDDYNAME)>

<cflog file="#CFEvent.GatewayID#Status" type="Information" text="in OnbuddyStatus, sender is #CFEvent.OriginatorID#">
<cflock scope="APPLICATION" timeout="10" type="EXCLUSIVE">
// Create the status structures if they don't exist.
if (NOT StructKeyExists(Application, "buddyStatus")) {
    Application.buddyStatus=StructNew();
}
if (NOT StructKeyExists(Application.buddyStatus, CFEvent.Data.BUDDYNAME)) {
    Application.buddyStatus[#CFEvent.Data.BUDDYNAME#]=StructNew();
}
// Save the buddy status, timestamp, and custom away message
Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].status=CFEvent.Data.BUDDYSTATUS;
Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].timeStamp=CFEvent.Data.TIMESTAMP;
// The following assumes that the buddy is in only one group.
Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].customAway=mybuddyinfo[1].BUDDYCUSTOMAWAYMESSAGE;
</cfscript>
</cflock>
<!--- log the info, for debugging purposes only --->
<cfset temp=Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].status>
<cflog file="#CFEvent.GatewayID#Status" type="Information" text="Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].status is #temp#">
<cfset temp=Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].timeStamp>
<cflog file="#CFEvent.GatewayID#Status" type="Information" text="Application.buddyStatus[#CFEvent.Data.BUDDYNAME#].timestamp is #temp#">
<cflog file="#CFEvent.GatewayID#Status" type="Information" text="Buddy Custom Away Message is mybuddyinfo[1].BUDDYCUSTOMAWAYMESSAGE#">
</cffunction>
onIMServerMessage

Description

Handles incoming error and status messages from the IM server.

Syntax

```plaintext
onIMServerMessage(CFEvent)
```

See also

onIncomingMessage, onAddBuddyRequest, onAddBuddyResponse, onBuddyStatus

Parameters

This method must take one parameter, a CFEvent structure with the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayType</td>
<td>Gateway type, either XMPP or SAMETIME</td>
</tr>
<tr>
<td>gatewayID</td>
<td>The ID of the gateway instance, as configured in ColdFusion Administrator</td>
</tr>
<tr>
<td>originatorID</td>
<td>The IM ID (buddy name) of the message originator</td>
</tr>
<tr>
<td>cfcMethod</td>
<td>This CFC method; by default, onIMServerMessage</td>
</tr>
<tr>
<td>data.MESSAGE</td>
<td>The message sent by the server</td>
</tr>
<tr>
<td>data.SENDER</td>
<td>The sender's ID; identical to the originatorID</td>
</tr>
<tr>
<td>data.RECIPIENT</td>
<td>The recipient's ID, as specified in the gateway's configuration file</td>
</tr>
<tr>
<td>data.TIMESTAMP</td>
<td>The date and time when the message was sent</td>
</tr>
</tbody>
</table>

Example

The following example logs the sender, message, and a timestamp when an IM server sends an error or status message:
<cffunction name="onIMServerMessage">
<!--- This function just logs the message. --->
<cfargument name="CFEvent" type="struct" required="YES">
<cflog file="#CFEvent.GatewayID#Status" text="onIMServerMEssage; SENDER: #CFEvent.OriginatorID#MESSAGE: #CFEvent.Data.MESSAGE# TIMESTAMP: #CFEvent.Data.TIMESTAMP#">
</cffunction>
onIncomingMessage

Description
Handles incoming instant messages from other users. Optionally returns a response to the message sender.

Syntax

```plaintext
onIncomingMessage(CFEvent)
```

See also

onAddBuddyRequest, onAddBuddyResponse, onBuddyStatus, onIMServerMessage, Handling incoming messages in the Developing ColdFusion Applications

Parameters

The method must take one parameter, a CFEvent structure with the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gatewayType</td>
<td>Gateway type, either XMPP or SAMETIME.</td>
</tr>
<tr>
<td>gatewayID</td>
<td>The ID of the Gateway instance as configured in ColdFusion Administrator.</td>
</tr>
<tr>
<td>originatorID</td>
<td>The IM ID of the message originator.</td>
</tr>
<tr>
<td>cfcMethod</td>
<td>This CFC method; by default, onIncomingMessage.</td>
</tr>
<tr>
<td>data.MESSAGE</td>
<td>The message that was received.</td>
</tr>
<tr>
<td>data.SENDER</td>
<td>The sender's ID; identical to the originatorID</td>
</tr>
<tr>
<td>data.RECIPIENT</td>
<td>The recipient's ID, as specified in the gateway's configuration file</td>
</tr>
<tr>
<td>data.TIMESTAMP</td>
<td>The date and time when the message was sent</td>
</tr>
</tbody>
</table>

Returns

The function can optionally return a value to send a response message. The return structure must contain the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>Normally omitted. You can also specify submit.</td>
</tr>
<tr>
<td>buddyID</td>
<td>ID to which to send the message. Normally, the value of the input parameter's Data.SENDER field.</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>message</td>
<td>The message contents.</td>
</tr>
</tbody>
</table>

**Example**

The following example shows a simple `onIncomingMessage` method that echoes a message back to the sender.

```coldfusion
cffunction name="onIncomingMessage">
  <cfargument name="CFEvent" type="struct" required="YES">
  <cfset input_mesg = CFEvent.data.MESSAGE>
  <cfset retValue = structNew()>
  <cfset retValue.command = "submit"  >
  <cfset retValue.buddyID = CFEvent.originatorID>
  <cfset retValue.message = "Message Received:" & input_mesg>
  <cfreturn retValue>
</cffunction>
```
**IM Gateway GatewayHelper class methods**

The GatewayHelper class returned by the CFML `GetGatewayHelper` function includes the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Method</th>
<th>Method</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>addBuddy</td>
<td>getDenyList</td>
<td>getStatusAsString</td>
<td>removeDeny</td>
</tr>
<tr>
<td>addDeny</td>
<td>getName</td>
<td>getStatusTimeStamps</td>
<td>removePermit</td>
</tr>
<tr>
<td>addPermit</td>
<td>getNickName</td>
<td>isOnline</td>
<td>setNickName</td>
</tr>
<tr>
<td>getBuddyInfo</td>
<td>getPermitList</td>
<td>numberOfMessagesReceived</td>
<td>setPermitMode</td>
</tr>
<tr>
<td>getBuddyList</td>
<td>getPermitMode</td>
<td>numberOfMessagesSent</td>
<td>setStatus</td>
</tr>
<tr>
<td>getCustomAwayMessage</td>
<td>getProtocolName</td>
<td></td>
<td>removeBuddy</td>
</tr>
</tbody>
</table>

[back to top](#)
IM gateway message sending commands

You use the `SendGatewayMessage` CFML function or the return value of a CFC listener method to send outgoing messages. The ColdFusion IM gateway accepts the following outgoing message commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>submit</td>
<td>(Default) Sends a normal message to another IM user.</td>
</tr>
<tr>
<td>accept</td>
<td>Accepts an add buddy request. Adds the buddy to the list of IDs that get your presence information and sends an acceptance message to the buddy ID.</td>
</tr>
<tr>
<td>decline</td>
<td>Declines an add buddy request and sends a rejection message to the buddy ID.</td>
</tr>
<tr>
<td>noact</td>
<td>Tells the gateway to take no action. The gateway logs a message that indicates that it took no action, and contains the gateway type, gateway ID, and buddy ID.</td>
</tr>
</tbody>
</table>

The message structure that you return in the gateway listener CFC function or use as the second parameter in the CFML `SendGatewayMessage` function can have the following fields. The table lists the fields and the commands in which they are used, and describes the field's use.

<table>
<thead>
<tr>
<th>Field</th>
<th>Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buddyID</td>
<td>All</td>
<td>The destination user ID</td>
</tr>
<tr>
<td>command</td>
<td>All</td>
<td>The command; defaults to submit if omitted</td>
</tr>
<tr>
<td>message</td>
<td>submit</td>
<td>A text message to send to the destination user</td>
</tr>
<tr>
<td>reason</td>
<td>accept, decline</td>
<td>A text description of the reason for the action or other message to send to the add buddy requestor</td>
</tr>
</tbody>
</table>

In typical use, a ColdFusion application uses the accept, decline, and noact commands in the return value of the `onAddBuddyRequest` method, and uses the submit command (or no command, because submit is the default command) in `SendGatewayMessage` CFML functions and the return value of the `onIncomingMessage` CFC method.

#back to top
addBuddy

Description

Adds a buddy to the buddy list for the gateway user ID and asks to have the IM server send messages with the buddy’s online presence state to the gateway.

Syntax

```plaintext
Boolean = addBuddy(name, nickname, group)
```

See also

getBuddyInfo, getBuddyList, removeBuddy, Using the GatewayHelper object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The unique instant messaging user name for the person about whom you want to receive periodic status messages.</td>
</tr>
<tr>
<td>nickname</td>
<td>The nickname that the application can use to refer to the user.</td>
</tr>
<tr>
<td>group</td>
<td>The name of the group you wish to add the user to in your Buddy List. If the group specified does not exist, it will be created. If the group parameter is the empty string, the gateway uses the General group.</td>
</tr>
</tbody>
</table>

Returns

True if the ID was added to the gateway's buddy list; False, otherwise.

Usage

This method adds the buddy to the buddy list for the gateway's ID and sends a subscription request (to automatically get presence information about the buddy's online status) to the remote buddy. It does not wait for a response from the buddy, so it returns True (and the gateway adds the buddy to the list) even if the buddy denies the subscription request. Use the listener CFC onAddBuddyResponse method to monitor the buddy's response. If the CFEvent.data.MESSAGE field value is decline, the listener method can call the gatewayHelper object removeBuddy method to remove the buddy from the buddy list.

Example

See GatewayHelper example in Using the GatewayHelper object the Developing ColdFusion Applications, which uses all GatewayHelper class methods.
addDeny

Description

Tells the IM server to add the specified user to the deny list for the gateway's user ID. If the gateway's permit mode value is DENY_SOME, the specified user cannot receive messages on the gateway's presence state.

Syntax

```text
Boolean = addDeny(name, nickname, group)
```

See also

addPermit, getDenyList, getPermitList, getPermitMode, removeDeny, removePermit, setPermitMode, Using the GatewayHelper object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The unique instant messaging user name for the person about whom you want to deny access to status messages.</td>
</tr>
<tr>
<td>nickname</td>
<td>The nickname that the application can use to refer to the user. Can be the empty string.</td>
</tr>
<tr>
<td>group</td>
<td>The name of the group that you want to add the user to in your buddy list. If the group specified does not exist, it is created. If the group parameter is the empty string, the gateway uses the General group.</td>
</tr>
</tbody>
</table>

Returns

True if the ID was added to the deny list; False, otherwise.

⚠️ Note

If the XMPP server does not support permission management, this function always returns False

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.
addPermit

Description

Tells the IM server to add the specified user to the permit list for the gateway’s user ID. If the gateway’s permit mode is PERMIT_SOME, the specified user receive messages on the gateway's presence state.

Syntax

| Boolean = addPermit(name, nickname, group) |

See also

addDeny, getDenyList, getPermitList, getPermitMode, removeDeny, removePermit, setPermitMode, Using the GatewayHelper object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The unique instant messaging user name for the person about whom you want to deny access to status messages.</td>
</tr>
<tr>
<td>nickname</td>
<td>The nickname that the application can use to refer to the user. Can be the empty string.</td>
</tr>
<tr>
<td>group</td>
<td>The name of the group you want to add the user to in your Buddy List. If the group specified does not exist, it is created. If the group parameter is the empty string, the gateway uses the General group.</td>
</tr>
</tbody>
</table>

Returns

True if the ID was added to the permit list; false, otherwise.

⚠️ Note

If the XMPP server does not support permission management, this function always returns False.

Example

See GatewayHelper example, in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.
getBuddyInfo

Description

Gets information about the specified user from the buddy list, deny list, and permit list.

Syntax

```java
array = getBuddyInfo(name)
```

See also

`addBuddy`, `getBuddyList`, `removeBuddy`, Using the GatewayHelper object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The unique instant messaging user name for the person about whom you want to get information.</td>
</tr>
</tbody>
</table>

Returns

An array of structures, with one structure for each information record found. The method finds one record for each group that the user belongs to in each of the lists (buddy, permit, deny) that contains the specified name. Each structure has the following fields. Some fields might not be meaningful for some IM protocols. If there is no information for a field, it is blank.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUDDYNAME</td>
<td>The user's unique ID.</td>
</tr>
<tr>
<td>BUDDYGROUP</td>
<td>The group to which the user belongs.</td>
</tr>
<tr>
<td>BUDDYNICKNAME</td>
<td>The nickname that you have assigned to the user.</td>
</tr>
<tr>
<td>BUDDYPROTOCOL</td>
<td>The instant messaging protocol. JABBER (for XMPP) or SAMETIME, or an empty string (if the server did not return a value).</td>
</tr>
<tr>
<td><strong>BUDDYSTATUS</strong></td>
<td>The user's presence state, can be any of the following:</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>• ONLINE</td>
</tr>
<tr>
<td></td>
<td>• OFFLINE</td>
</tr>
<tr>
<td></td>
<td>• AWAY</td>
</tr>
<tr>
<td></td>
<td>• DND (displays as DO NOT DISTURB)</td>
</tr>
<tr>
<td></td>
<td>• NA (displays as NOT AVAILABLE)</td>
</tr>
<tr>
<td></td>
<td>• FREE_TO_CHAT (displays as FREE TO CHAT)</td>
</tr>
<tr>
<td></td>
<td>• IDLE</td>
</tr>
<tr>
<td></td>
<td><strong>XMPP only</strong></td>
</tr>
<tr>
<td></td>
<td>• NA (displays as NOT AVAILABLE)</td>
</tr>
<tr>
<td></td>
<td>• FREE_TO_CHAT (displays as FREE TO CHAT)</td>
</tr>
<tr>
<td></td>
<td>• IDLE</td>
</tr>
<tr>
<td></td>
<td><strong>Sametime only</strong></td>
</tr>
<tr>
<td></td>
<td>• IDLE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>BUDDYSIGNONTIME</strong></th>
<th>The date and time when the user signed onto the IM server. Empty if the user is not currently signed on. Always an empty string for XMPP and Sametime.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>BUDDYSTATUSTIME</strong></th>
<th>The date and time when the user's status most recently changed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>BUDDYCUSTOMAWAYMESSAGE</strong></th>
<th>The custom away message that the user has set to explain the current status, if any.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>BUDDYOWNER</strong></th>
<th>A string representing the client and protocol associated with this ID, in the format <code>client@protocol</code>.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>BUDDYLISTTIYPE</strong></th>
<th>The type of list that this buddy record is in; one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• BUDDY_LIST - The list of users whose presence status information the gateway can receive.</td>
</tr>
<tr>
<td></td>
<td>• DENY_LIST - The list of users who cannot get presence information about the gateway ID.</td>
</tr>
<tr>
<td></td>
<td>• PERMIT_LIST - The list of users who can send presence information messages to the gateway ID.</td>
</tr>
<tr>
<td></td>
<td>• REVERSE_LIST - The list of users who do not allow messages to us.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>BUDDYIDLETIME</strong></th>
<th>If the buddy status is IDLE, how long the buddy has been idle. Always 0 for XMPP or SameTime.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>BUDDYISMOMOBILE</strong></th>
<th>True or False, indicating whether the user is on a mobile device. Always False for XMPP or SameTime.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>BUDDYWARNINGPERCENT</strong></th>
<th>The user's warning percentage value. Always 0 for XMPP or SameTime.</th>
</tr>
</thead>
</table>

Example

© 2014 Adobe Systems Incorporated. All rights reserved. 2860
See Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods. For an example of using this method to get the buddy custom away message, see onBuddyStatus.

#back to top
getBuddyList

Description

Gets the buddy list for the gateway's user ID.

Syntax

```
array = getBuddyList()
```

See also

addBuddy, getBuddyInfo, removeBuddy, Using the GatewayHelper object in the Developing ColdFusion Applications

Returns

An array of IDs (buddy names) of the users on the gateway's buddy list, a list of instant messaging IDs that this gateway normally communicates with.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications which uses all GatewayHelper class methods.

#back to top
getCustomAwayMessage

Description

Returns the gateway's custom away message if it has been set by the gatewayHelper object setStatus method.

Syntax

```
string = getCustomAwayMessage()
```

See also

`getStatusAsString`, `getStatusTimeStamp`, `isOnline`, `setStatus`, *Using the GatewayHelper object* in the Developing ColdFusion Applications

Returns

The gateway's custom away message if it has been set by the GatewayHelper object setStatus method.

Example

See GatewayHelper example in *Using the GatewayHelper object* in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

#back to top
**getDenyList**

**Description**

Returns the list of users that the IM server has been told not to send state information about the gateway, if the permit mode is set to DENY_SOME.

**Syntax**

```java
array = getDenyList()
```

**See also**

`addDeny`, `addPermit`, `getPermitList`, `getPermitMode`, `removeDeny`, `removePermit`, `setPermitMode`, `Using the GatewayHelper object` in the *Developing ColdFusion Applications*

**Returns**

An array of IDs (buddy names) of the users on the gateway's deny list, the list of IDs to which the IM server does not send presence status information.

⚠️ **Note**

If the XMPP server does not support permission management, this function always returns False.

**Example**

See `GatewayHelper example` in *Using the GatewayHelper object* in the *Developing ColdFusion Applications*, which uses all GatewayHelper class methods.

#back to top
getName

Description

Returns the gateway's user name.

Syntax

```
string = getName()
```

See also

getProtocolName, numberOfMessagesReceived, numberOfMessagesSent, setNickName. Using the GatewayHelper object in the Developing ColdFusion Applications

Returns

The gateway's user name, as specified in gateway configuration file.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

#back to top
**getNickName**

**Description**

Returns the gateway’s nickname (display name), if it has been set using the gatewayHelper object `setNickName` method.

**Syntax**

```
string = getNickName()
```

**See also**

`getName`, `getProtocolName`, `numberOfMessagesReceived`, `numberOfMessagesSent`, `setNickName`, `Using the GatewayHelper object` in the *Developing ColdFusion Applications*

**Returns**

The gateway’s nickname, if any; empty string, otherwise.

**Example**

See *GatewayHelper example* in `Using the GatewayHelper object` in the *Developing ColdFusion Applications*, which uses all GatewayHelper class methods.

*back to top*
getPermitList

Description

Returns the list of users that the IM server has been told to send state information about the gateway.

Syntax

```plaintext
array = getPermitList()
```

See also

addDeny, addPermit, getDenyList, getPermitMode, removeDeny, removePermit, setPermitMode, Using the GatewayHelper object in the Developing ColdFusion Applications

Returns

An array of IDs (buddy names) of the users on the gateway’s permit list, the list of IDs to which the IM server sends presence status information if the permit mode is set to PERMIT_SOME.

⚠️ Note

If the XMPP server does not support permission management, this function always returns False.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

#back to top
getPermitMode

Description

Gets the gateway's permit mode from the IM server. The permit mode determines whether all users can get the gateway's online state information, or whether the server uses a permit list or a deny list to control which users get state information.

Syntax

```
string = getPermitMode()
```

See also

addDeny, addPermit, getDenyList, getPermitList, removeDeny, removePermit, setPermitMode, Using the GatewayHelper object in Developing ColdFusion Applications

Returns

The gateway's permit mode; one of the following values:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERMIT_ALL</td>
<td>(Default) Permits all users to be aware of the gateway's online presence and state.</td>
</tr>
<tr>
<td>PERMIT_SOME</td>
<td>Permits only users in the permit list to be aware of the gateway's online presence and state.</td>
</tr>
<tr>
<td>DENY_SOME</td>
<td>Prevents the users in the deny list from being aware of the gateway's online presence and state.</td>
</tr>
</tbody>
</table>

⚠️ Note

If the XMPP server does not support permission management, this function always returns PERMIT_ALL.

Example

See GatewayHelper example in Using the GatewayHelper object in Developing ColdFusion Applications, which uses all GatewayHelper class methods.

#back to top
getProtocolName

Description

Gets the name of the gateway's instant messaging protocol.

Syntax

```java
string = getProtocolName()
```

See also

getName, getNickName, numberOfMessagesReceived, numberOfMessagesSent, setNickName. Using the GatewayHelper object in the Developing ColdFusion Applications

Returns

The gateway's protocol, as determined by the gateway type; one of the following values:

- JABBER (for XMPP)
- SAMETIME

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.
getStatusAsString

Description

Gets the online status of the gateway as a text string.

Syntax

```coldfusion
string = getStatusAsString()
```

See also

getStatusTimeStamp, isOnline, setStatus, Using the GatewayHelper object in the Developing ColdFusion Applications

Returns

The gateway's online status; one of the following:

- ONLINE
- OFFLINE
- AWAY
- DO NOT DISTURB
- NOT AVAILABLE
- FREE TO CHAT
- IDLE

Usage

The DO NOT DISTURB, NOT AVAILABLE, and FREE TO CHAT strings differ from the status values that you use in the setStatus method, which does not allow spaces in the status names.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

#back to top
getStatusTimeStamp

Description

Gets the date and time that the gateway changed its online status.

Syntax

```
date-time object = getStatusTimeStamp()
```

See also

getStatusAsString, isOnline, setStatus, Using the GatewayHelper object in the Developing ColdFusion Applications

Returns

The date and time that the gateway changed its online status, normally by calling the setStatus gatewayHelper object method.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

#back to top
**numberOfMessagesReceived**

**Description**

Gets the number of messages received by the gateway since it was started.

**Syntax**

```java
integer = numberOfMessagesReceived()
```

**See also**

`getName`, `getNickName`, `getProtocolName`, `numberOfMessagesSent`, `setNickName`, Using the GatewayHelper object in Developing ColdFusion Applications

**Returns**

The number of messages received by the gateway since it was started.

**Example**

See GatewayHelper example in Using the GatewayHelper object in Developing ColdFusion Applications, which uses all GatewayHelper class methods.

[#back to top](#)
**numberOfMessagesSent**

**Description**

Gets the number of messages sent by the gateway since it was started.

**Syntax**

```java
integer = numberOfMessagesSent()
```

**See also**

`getName`, `getNickName`, `getProtocolName`, `numberOfMessagesReceived`, `setNickName`, *Using the GatewayHelper object* in the *Developing ColdFusion Applications*

**Returns**

The number of messages sent by the gateway since it was started.

**Example**

See  *GatewayHelper example* in *Using the GatewayHelper object* in the *Developing ColdFusion Applications*, which uses all GatewayHelper class methods.

*back to top*
removeBuddy

Description
Removes an ID from a group in the buddy list for the gateway and tells the IM server not to send the gateway messages with the buddy's online presence state.

Syntax

```csharp
Boolean = removeBuddy(name, group)
```

See also

addBuddy, getBuddyInfo, getBuddyList, removeDeny, removePermit, Using the GatewayHelper object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The unique instant messaging user name for the person to remove from the buddy list.</td>
</tr>
<tr>
<td>group</td>
<td>The name of the group from which you want to remove the user. If the parameter is the empty string, the gateway uses the General group.</td>
</tr>
</tbody>
</table>

Returns

True if the ID was removed from the group; False, otherwise.

Usage

If the user is in multiple groups in your buddy list, you remove the buddy separately from each group. The IM server does not stop sending status updates until you remove the name from all groups.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.
removeDeny

Description

Removes an ID from a group in the deny list for the gateway. If the gateway's permit mode is DENY_SOME, the specified user can receive messages on the gateway's presence state.

Syntax

```java
Boolean = removeDeny(name, group)
```

See also

addDeny, addPermit, getDenyList, getPermitList, getPermitMode, removeBuddy, removePermit, setPermitMode, Using the GatewayHelper object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The unique instant messaging user name for the person to remove from the deny list.</td>
</tr>
<tr>
<td>group</td>
<td>The name of the group from which you want to remove the user. If the parameter is the empty string, the gateway uses the General group.</td>
</tr>
</tbody>
</table>

Returns

True if the ID was removed from the group; False, otherwise.

⚠️ Note

If the XMPP server does not support permission management, this function always returns False.

Usage

If the user is in multiple groups in your deny list, you remove the user separately from each group. The IM server enables sending status updates if you remove the name any group.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

#back to top
removePermit

Description

Removes an ID from a group in the permit list for the gateway. If the gateway's permit mode is PERMIT_SOME, the specified user cannot receive messages on the gateway's presence state.

Syntax

```
Boolean = removePermit(name, group)
```

See also

addDeny, addPermit, getDenyList, getPermitList, getPermitMode, removeBuddy, removeDeny, setPermitMode, Using the GatewayHelper object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The unique instant messaging user name for the person to remove from the permit list.</td>
</tr>
<tr>
<td>group</td>
<td>The name of the group from which you want to remove the user. If the parameter is the empty string, the gateway uses the General group.</td>
</tr>
</tbody>
</table>

Returns

True if the ID was removed from the group; False, otherwise.

⚠️ Note

If the XMPP server does not support permission management, this function always returns False.

Usage

If the user is in multiple groups in your permit list, you remove the user separately from each group. However, the IM server stops sending status updates when you remove the user from the first group.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

#back to top
setNickName

Description
Sets the gateway's nickname (display name).

Syntax

```
Boolean = setNickName(name)
```

See also

gName, getNickName, getProtocolName, numberOfMessagesReceived, numberOfMessagesSent, Using the GatewayHelper object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The display name that you want to associate with this gateway. This name is not guaranteed to be unique for the protocol.</td>
</tr>
</tbody>
</table>

Returns

True if the nickname got set; false, otherwise.

Example

See GatewayHelper example in Using the GatewayHelper object in the Developing ColdFusion Applications, which uses all GatewayHelper class methods.

#back to top
setPermitMode

Description
Sets the gateway's permit mode on the IM server. The permit mode determines whether all users can get the gateway's online state information, or whether the server uses a permit list or a deny list to control which users get state information.

Syntax

```
Boolean = setPermitMode(permitMode)
```

See also

`addDeny, addPermit, getDenyList, getPermitList, getPermitMode, removeDeny, removePermit, Using the GatewayHelper object` in the `Developing ColdFusion Applications`

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>permitMode</td>
<td>The permission mode, one of the following:</td>
</tr>
<tr>
<td></td>
<td>• PERMIT_ALL-Permits all users to be aware of the gateway's online presence and state. This is the default mode if you do not call this function.</td>
</tr>
<tr>
<td></td>
<td>• PERMIT_SOME-Permits only users in the permit list to be aware of the gateway's online presence and state.</td>
</tr>
<tr>
<td></td>
<td>• DENY_SOME-Prevents all users in the deny list from being aware of the gateway's online presence and state.</td>
</tr>
</tbody>
</table>

Returns

True if the permit mode was set; False otherwise.

⚠️ Note

If the XMPP server does not support permission management, this function returns False to all values except PERMIT_ALL.

Example

See `GatewayHelper example` in `Using the GatewayHelper object` in the `Developing ColdFusion Applications`, which uses all GatewayHelper class methods.

#back to top
setStatus

Description
Sets the online presence status of the gateway, including any custom away message.

Syntax

```
Boolean = setStatus(status, customAwayMsg)
```

See also

`getCustomAwayMessage`, `getStatusAsString`, `getStatusTimeStamp`, `isOnline`, Using the GatewayHelper object in the Developing ColdFusion Applications

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>The gateway's online presence status; one of the following:</td>
</tr>
<tr>
<td></td>
<td>• ONLINE</td>
</tr>
<tr>
<td></td>
<td>• AWAY</td>
</tr>
<tr>
<td></td>
<td>• DND (Do Not Disturb)</td>
</tr>
<tr>
<td></td>
<td>• NA (Not Available)</td>
</tr>
<tr>
<td></td>
<td>• FREE_TO_CHAT</td>
</tr>
<tr>
<td></td>
<td>• IDLE</td>
</tr>
<tr>
<td></td>
<td><strong>XMPP only</strong></td>
</tr>
<tr>
<td></td>
<td>• NA (Not Available)</td>
</tr>
<tr>
<td></td>
<td>• FREE_TO_CHAT</td>
</tr>
<tr>
<td></td>
<td>• IDLE</td>
</tr>
<tr>
<td></td>
<td><strong>Sametime only:</strong></td>
</tr>
<tr>
<td></td>
<td>• IDLE</td>
</tr>
<tr>
<td>customAwayMsg</td>
<td>A text string containing a custom message for the status. Can be the empty string if you do not need a custom away message.</td>
</tr>
</tbody>
</table>

Returns

True, if the operation was successful; False. otherwise. Passing an invalid status for the protocol causes this method to return False.

Usage
Do not use the `setStatus` method to go offline. Although the method accepts a parameter of `OFFLINE`, the gateway immediately resets itself to be online. To set the gateway offline, stop the gateway instance in the ColdFusion Administrator, or use the `stopGatewayInstance` method in the CFIDE.adminapi.eventgateway CFC.

Example

See *GatewayHelper example* in [Using the GatewayHelper object](#) in the *Developing ColdFusion Applications*, which uses all GatewayHelper class methods.
SMS Gateway CFEvent structure and commands

This section describes the detailed contents of the following structures that you use in the SMS Gateway listener CFCs and CFML SendGatewayMessage functions.

#back to top
### SMS Gateway incoming message CFEvent structure

The SMS gateway puts the following information in a CFEvent instance that it sends to the CFC listener method:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OriginatorID</td>
<td>Contents of the PDU <code>source_addr</code> field, the address of the device that sent the message.</td>
</tr>
<tr>
<td>CfcMethod</td>
<td>Listener CFC method name. Value of the configuration file <code>cfc-method</code> entry, or <code>onIncomingMessage</code> if the configuration file does not have this entry.</td>
</tr>
<tr>
<td>Data.MESSAGE</td>
<td>Contents of the <code>short_message</code> field of the PDU.</td>
</tr>
<tr>
<td>Data.sourceAddress</td>
<td>The address of the device that sent this message.</td>
</tr>
<tr>
<td>Data.destAddress</td>
<td>The address to which the message was sent; an address in the range specified by the gateway configuration file <code>address-range</code> setting.</td>
</tr>
<tr>
<td>Data.esmClass</td>
<td>Contents of the PDU <code>esm_class</code> field. Identifies the message type. A number in the range 0-255 representing a Byte value, where bits 2-5 (0-indexed) indicate the message type, and therefore the contents of the data.MESSAGE field, as follows. (Reserved values are omitted.)xx0000xx-Normal message{{xx0001xx}}-SMSC delivery receipt{{xx0010xx}}SME Delivery Acknowledgement{{xx0100xx}}-SME Manual/User Acknowledgement{{xx0110xx}}-Conversation abort (Korean CDMA only)xx1000xx-Intermediate Delivery NotificationFor more information on this field, see the SMPP specification.</td>
</tr>
<tr>
<td>Data.protocol</td>
<td>Contents of the PDU <code>protocol_id</code> field. Meaningful for messages sent from GSM networks only. For more information, see the GSM 03.40 specification.</td>
</tr>
<tr>
<td>Data.priority</td>
<td>Contents of the PDU <code>priority_flag</code> field. A message priority level set by the originating SME, in the range 0-3; 0 is the lowest priority and 3 is the highest priority. The specific priority level meaning depends on the originating network. For more details, see the SMPP specification.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Data.registeredDelivery</td>
<td>Contents of the PDU registered_delivery field, indicating the type of delivery receipt or acknowledgement that the sender requested. A number in the range 0-32, representing the sum of the following values: 0: No SMS delivery receipt requested or 1: SMSC delivery receipt requested on delivery success or failure or 2: SMSC delivery receipt requested on delivery failure only Plus 0: No SME acknowledgement requested or 4: SME Delivery Acknowledgement requested or 8: SME Manual/User Acknowledgement requested or 12: Both Delivery and Manual/User Acknowledgements requested Plus 0: No Intermediate notification requested or 16: Intermediate notification requested</td>
</tr>
<tr>
<td>Data.DataCoding</td>
<td>Contents of the PDU data_coding field. Indicates the character set or the noncharacter data type of the message contents, as follows: 00000000--SMSC Default Alphabet 00000001--IA5 (CCITT T.50)/ASCII (ANSI X3.4) 00000010--Octet unspecified (8-bit binary) 00000011--Latin 1 (ISO-8859-1) 00000100--Octet unspecified (8-bit binary) 00000101--JIS (X 0208-1990) 00000110--Cyrillic (ISO-8859-5) 00000111--Latin/Hebrew (ISO-8859-8) 00001000--UCS2 (ISO/IEC-10646) 00001001--Pictogram Encoding 00001010--ISO-2022-JP (Music Codes) 00001100--Extended Kanji JIS(X 0212-1990) 00001110--KS C 560111xxxxxx--GSM control use only; see the GSM 03.38 specification For more details, see the SMPP specification.</td>
</tr>
<tr>
<td>Data.messageLength</td>
<td>The length of the short_message field.</td>
</tr>
<tr>
<td>GatewayType</td>
<td>Always SMS.</td>
</tr>
</tbody>
</table>

For more information on the meanings of some of these fields and how to handle incoming SMS messages an SMS gateway listener CFC method, see [Handling incoming messages](#) in the Developing ColdFusion Applications.
SMS gateway message sending commands

ColdFusion applications that use gateways of the Short Message Service (SMS) type can send the commands (described in this section) to the event gateway in an outgoing message.
submit command

To send a message to a single destination address in an SMPP SUBMIT_SM PDU, the structure that you used in the *Data* parameter of a *SendGatewayMessage* function or the return variable of the CFC listener method has the following fields. For more information about these fields, see the documentation for the SUBMIT_MULTI PDU in the SMPP3.4 specification, which you can download from the SMS Forum at [www.smsforum.net/](http://www.smsforum.net/).

**Required fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>command</strong></td>
<td>If present, the value must be <em>submit</em>. If you omit this field, the event gateway sends a submit message.</td>
</tr>
<tr>
<td><strong>shortMessage</strong> or <strong>messagePayload</strong></td>
<td>The message contents. You must specify one of these fields, but not both. The SMPP specification imposes a maximum size of 254 bytes on the shortMessage field, and some carriers might limit its size further. The messagePayload field can contain up to 64K bytes; it must start with 0x0424, followed by two bytes specifying the payload length, followed by the message contents.</td>
</tr>
<tr>
<td><strong>destAddress</strong></td>
<td>Required. The address to which to send the message.</td>
</tr>
<tr>
<td><strong>sourceAddress</strong></td>
<td>The address of this application. You can omit this field; the configuration file specifies the application address.</td>
</tr>
</tbody>
</table>

**Optional fields**

You can set default values for the following optional fields in the SMS event gateway configuration file. For information on the default values, see [Configuring an SMS event gateway](http://www.adobe.com/cfappdev/guides/cffusion_development.html) in the *Developing ColdFusion Applications*.

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>destAddress_npi</strong></td>
<td></td>
</tr>
<tr>
<td><strong>destAddress_ton</strong></td>
<td></td>
</tr>
<tr>
<td><strong>serviceType</strong></td>
<td></td>
</tr>
</tbody>
</table>

The following optional fields do not have default values:

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>alertOnMsgDelivery</strong></td>
<td>EsmClass</td>
</tr>
<tr>
<td><strong>priorityFlag</strong></td>
<td>smDefaultMsgId</td>
</tr>
<tr>
<td><strong>callbackNum</strong></td>
<td>ItsReplyType</td>
</tr>
<tr>
<td><strong>PrivacyIndicator</strong></td>
<td>SmsSignal</td>
</tr>
<tr>
<td><strong>callbackNumAtag</strong></td>
<td>ItsSessionInfo</td>
</tr>
<tr>
<td><strong>protocolId</strong></td>
<td>SourceAddrSubunit</td>
</tr>
<tr>
<td><strong>callbackNumPresInd</strong></td>
<td>LanguageIndicator</td>
</tr>
<tr>
<td><strong>registeredDelivery</strong></td>
<td>SourcePort</td>
</tr>
<tr>
<td><strong>dataCoding</strong></td>
<td>MoreMsgsToSend</td>
</tr>
<tr>
<td><strong>replaceIfPresent</strong></td>
<td>SourceSubaddress</td>
</tr>
</tbody>
</table>
Example

The following example `onIncomingMessage` method of a listener CFC uses the submit command to echo incoming SMS messages to the message originator:

```coldfusion
<cffunction name="onIncomingMessage" output="no">
  <cfargument name="CFEvent" type="struct" required="yes">
  <!--- Create a return structure that contains the message. --->
  <cfset retValue = structNew()>
  <cfset retValue.command = "submit">
  <cfset retValue.destAddress = arguments.CFEvent.originatorid>
  <cfset retValue.shortMessage = "Echo: " & CFEvent.Data.MESSAGE>
  <!--- Send the message back. --->
  <cfreturn retValue>
</cffunction>
```

#back to top
submit Multi command

To send a single text message to multiple recipients using an SMPP SUBMIT_MULTI PDU, the Data parameter of a SendGatewayMessage function or the return variable of the CFC listener method usually has the following fields. For more information about these fields, see the documentation for the SUBMIT_MULTI PDU in the SMPP3.4 specification, which you can download from the SMS Forum at www.smsforum.net/.

Required fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>Must be submitMulti.</td>
</tr>
<tr>
<td>shortMessage or messagePayload</td>
<td>The message contents. You must specify one of these fields, but not both. The SMPP specification imposes a maximum size of 254 bytes on the shortMessage field, and some carriers might limit its size further. The messagePayload field can contain up to 64K bytes; it must start with 0x0424, followed by two bytes specifying the payload length, followed by the message contents.</td>
</tr>
<tr>
<td>destAddress</td>
<td>A ColdFusion array of destination addresses (required). You cannot specify individual TON and NPI values for these addresses; all must conform to a single setting.</td>
</tr>
<tr>
<td>sourceAddress</td>
<td>The address of this application. You can omit this field; the configuration file specifies the application address.</td>
</tr>
</tbody>
</table>

Optional fields

The following optional fields can have default values set in the SMS event gateway configuration file. For information on the default values see Configuring an SMS event gateway in the Developing ColdFusion Applications.

<table>
<thead>
<tr>
<th>destAddress_npi</th>
<th>destAddress_ton</th>
<th>serviceType</th>
</tr>
</thead>
</table>

The following optional fields do not have default values:

<table>
<thead>
<tr>
<th>alertOnMsgDelivery</th>
<th>DisplayTime</th>
<th>protocolId</th>
<th>SmsSignal</th>
</tr>
</thead>
<tbody>
<tr>
<td>callbackNum</td>
<td>EsmClass</td>
<td>registeredDelivery</td>
<td>SourceAddrSubunit</td>
</tr>
<tr>
<td>callbackNumAtag</td>
<td>LanguageIndicator</td>
<td>replaceIfPresent</td>
<td>SourcePort</td>
</tr>
<tr>
<td>callbackNumPresInd</td>
<td>MsMsgWaitFacilities</td>
<td>SarMsgRefNum</td>
<td>SourceSubaddress</td>
</tr>
<tr>
<td>dataCoding</td>
<td>MsValidity</td>
<td>SarSegmentSeqnum</td>
<td>UserMessageReferenc e</td>
</tr>
</tbody>
</table>
Example

The following example `onIncomingMessage` method sends a response that echoes an incoming message to the originator address, and sends a copy of the response to a second address:

```cfm
<cffunction name="onIncomingMessage" output="no">
  <cfargument name="CFEvent" type="struct" required="yes">
  <!--- Get the message. --->
  <cfset data=cfevent.DATA>
  <cfset message="#data.message#">
  <!--- Create the return structure. --->
  <cfset retValue = structNew()>
  <cfset retValue.command = "submitmulti">
  <cfset retValue.destAddresses=arraynew(1)>
  <!--- One destination is incoming message originator; get the address from CFEvent originator ID. --->
  <cfset retValue.destAddresses[1] = arguments.CFEvent.originatorid>
  <cfset retValue.destAddresses[2] = "12345">
  <cfset retValue.shortMessage = "echo: " & message>
  <creturn retValue>
</cffunction>
```

#back to top
**data command**

To send binary data to a single destination address in an SMPP DATA_SM PDU, the `Data` parameter of a `SendGatewayMessage` function or the return variable of the CFC listener method must have the following fields. For more information about these fields, see the documentation for the SUBMIT_MULTI PDU in the SMPP3.4 specification, which you can download from the SMS Forum at [www.smsforum.net/](http://www.smsforum.net/).

**Required fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>Must be <code>data</code>.</td>
</tr>
<tr>
<td>messagePayload</td>
<td>The message data. To convert data to binary format, use the ColdFusion <code>ToBinary</code> function.</td>
</tr>
<tr>
<td>destAddress</td>
<td>The address to which to send the message.</td>
</tr>
<tr>
<td>sourceAddress</td>
<td>The address of this application. You can omit this field; the configuration file specifies the application address.</td>
</tr>
</tbody>
</table>

**Optional fields**

The following optional fields can have default values set in the SMS event gateway configuration file. For information on the default values see Configuring an SMS event gateway in the Developing ColdFusion Applications.

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>destAddress_npl</td>
<td>destAddress_ton</td>
</tr>
<tr>
<td>serviceType</td>
<td></td>
</tr>
</tbody>
</table>

The following optional fields do not have default values:

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>alertOnMsgDelivery</td>
<td>DestTelematicsId</td>
</tr>
<tr>
<td></td>
<td>NetworkErrorCode</td>
</tr>
<tr>
<td></td>
<td>SetDpf</td>
</tr>
<tr>
<td>callbackNum</td>
<td>DisplayTime</td>
</tr>
<tr>
<td></td>
<td>NumberOfMessages</td>
</tr>
<tr>
<td></td>
<td>SmsSignal</td>
</tr>
<tr>
<td>callbackNumAtag</td>
<td>EsmClass</td>
</tr>
<tr>
<td></td>
<td>PayloadType</td>
</tr>
<tr>
<td></td>
<td>SourceAddrSubunit</td>
</tr>
<tr>
<td>callbackNumPresInd</td>
<td>ItsReplyType</td>
</tr>
<tr>
<td></td>
<td>PrivacyIndicator</td>
</tr>
<tr>
<td></td>
<td>SourceBearerType</td>
</tr>
<tr>
<td>dataCoding</td>
<td>ItsSessionInfo</td>
</tr>
<tr>
<td></td>
<td>QosTimeToLive</td>
</tr>
<tr>
<td></td>
<td>SourceNetworkType</td>
</tr>
<tr>
<td>DestAddrSubunit</td>
<td>LanguageIndicator</td>
</tr>
<tr>
<td></td>
<td>ReceivedMessgeId</td>
</tr>
<tr>
<td></td>
<td>SourcePort</td>
</tr>
<tr>
<td>DestBearerType</td>
<td>MessageState</td>
</tr>
<tr>
<td></td>
<td>registeredDelivery</td>
</tr>
<tr>
<td></td>
<td>SourceSubaddress</td>
</tr>
<tr>
<td>DestNetworkType</td>
<td>MoreMsgsToSend</td>
</tr>
<tr>
<td></td>
<td>SarMsgRefNum</td>
</tr>
<tr>
<td></td>
<td>SourceTelematicsId</td>
</tr>
<tr>
<td>DestinationPort</td>
<td>MsMsgWaitFacilities</td>
</tr>
<tr>
<td></td>
<td>SarSegmentSegnum</td>
</tr>
<tr>
<td></td>
<td>UserMessageReferenc e</td>
</tr>
</tbody>
</table>
Example

The following example `onIncomingMessage` method converts an incoming message to binary data, and sends the binary version of the message back to the originator address:

```cfc
<cffunction name="onIncomingMessage" output="no">
  <cfargument name="CFEvent" type="struct" required="yes">
    <!--- Get the message --->
    <cfset data=CFEvent.DATA>
    <cfset message="#data.message#">
    <!--- Create the return structure --->
    <cfset retValue = structNew()>
    <!--- Sending to incoming message originator; get value from CFEvent. --->
    <cfset retValue.destAddress = arguments.CFEvent.originatorid>
    <cfset retValue.messagePayload = tobinary(tobase64("echo: " & message))>
    <cfreturn retValue>
</cffunction>
```

#back to top
CFML event gateway SendGatewayMessage data parameter

The ColdFusion CFML gateway type enables you to invoke CFC methods asynchronously. The structure that you use in the `SendGatewayMessage` function `data` parameter can include two types of fields:

- Any number of fields can contain arbitrary contents for use in by the CFC.
- Several optional fields can configure how the gateway delivers the information to the CFC. The CFML gateway looks for the following optional fields, and, if they exist, uses them to determine how it delivers the message. Do not use these field names for data that you send to your CFC method.

<table>
<thead>
<tr>
<th>Field</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cfcpath</code></td>
<td>Overrides the CFC path specified in the ColdFusion Administrator. This field lets you use a single gateway configuration in the ColdFusion Administrator multiple CFCs. This field sets the CFEvent object CFCPath variable.</td>
</tr>
<tr>
<td><code>method</code></td>
<td>Specifies the name of the method to invoke in the CFC. The default method is <code>onIncomingMessage</code>. This field lets you use a single gateway configuration in the ColdFusion Administrator for a CFC that has several methods. This field sets the CFEvent object CFCMethod variable.</td>
</tr>
<tr>
<td><code>originatorID</code></td>
<td>Sets the originatorID field of the CFEvent object that ColdFusion delivers to the CFC. The default value is CFMLGateway.</td>
</tr>
<tr>
<td><code>timeout</code></td>
<td>Sets the time-out, in seconds, during which the listener CFC must process the event request and return before ColdFusion gateway services terminates the request. The default value is the Timeout Request value set on the Server Settings page in the ColdFusion Administrator. Set this value if a request might validly take longer to process than the default time-out; for example, if the request involves a long processing time. This field sets the CFEvent object CFCTimeout variable.</td>
</tr>
</tbody>
</table>

**Example**

The following example consists of a CFML page that sends a message to a logevent method in the file logger.CFC. The CFML page specifies the CFC and method to call, and sets the OriginatorID.
<h3>Sending an event using a generic CFML event gateway and specifying the CFC and method.</h3>
<cfscript>
status = False;
props = structNew();
props.cfcpath="C:\CFusionMX7\gateway\cfc\MyCFCs\logger.cfc";
props.method="logEvent";
props.OriginatorID=CGI.SCRIPT_NAME;
props.Message="Replace me with a variable with data to log";
props.file="GenericCFCtest";
props.type="warning";
status = SendGatewayMessage("DefaultCFC", props);
if (status IS True)
  WriteOutput('Event Message "#props.Message#" has been sent.');
</cfscript>

The CFC method uses the OriginatorID and the message, file, and type fields of the CFEvent parameter's data field to specify the log file and message.

<cfcomponent>
<cffunction name="logEvent" output="no">
  <cfargument name="CFEvent" type="struct" required="yes">
    <cfscript>
      if (NOT IsDefined("CFEvent.Data.file")) {
        CFEvent.Data.file="defaultEventLog";
      }
      if (NOT IsDefined("CFEvent.Data.type")) {
        CFEvent.Data.type="information";
      }
    </cfscript>
    <cflog text="Message from #CFEvent.originatorID#: #CFEvent.Data.message#" file="#CFEvent.data.file#" type="#CFEvent.Data.type#" />
  </cffunction>
</cfcomponent>

- Example
ColdFusion C++ CFX Reference

ColdFusion includes CFXAPI classes and methods for building ColdFusion extensions.

- C++ class overview
- Deprecated class methods
- CCFXException class
- CCFXQuery class
- CCFXRequest class
- CCFXStringSet class

#back to top
### C++ class overview

The following table lists the CFXAPI classes and methods:

<table>
<thead>
<tr>
<th>Class</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCFXException class</td>
<td>:GetError&lt;br&gt;:GetDiagnostics</td>
</tr>
<tr>
<td>CCFXQuery class</td>
<td>:AddRow&lt;br&gt;:GetColumns&lt;br&gt;:GetData&lt;br&gt;:GetName&lt;br&gt;:GetRowCount&lt;br&gt;:GetRowCount&lt;br&gt;:SetData</td>
</tr>
<tr>
<td>CCFXRequest class</td>
<td>:AddQuery&lt;br&gt;:AttributeExists&lt;br&gt;:CreateStringSet&lt;br&gt;:Debug&lt;br&gt;:GetAttribute&lt;br&gt;:GetAttributeList&lt;br&gt;:GetCustomData&lt;br&gt;:GetQuery&lt;br&gt;:ReThrowException&lt;br&gt;:SetCustomData&lt;br&gt;:SetVariable&lt;br&gt;:ThrowException&lt;br&gt;:Write&lt;br&gt;:WriteDebug</td>
</tr>
<tr>
<td>CCFXStringSet class</td>
<td>:AddString&lt;br&gt;:GetCount&lt;br&gt;:GetIndexForString&lt;br&gt;:GetString</td>
</tr>
</tbody>
</table>
# Deprecated class methods

The following CFXAPI classes and methods are deprecated. They do not work, and might cause an error, in later releases.

<table>
<thead>
<tr>
<th>Class</th>
<th>Deprecated member</th>
<th>Deprecated as of this ColdFusion release</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCFXQuery Class</td>
<td>CCFXQuery::SetQueryStringCCFXQuery::SetTotalTime</td>
<td>ColdFusion MX</td>
</tr>
<tr>
<td>CCFXRequest Class</td>
<td>CCFXRequest::GetSetting</td>
<td>ColdFusion MX</td>
</tr>
</tbody>
</table>

[back to top]
CCFXException class

An abstract class that represents an exception thrown during processing of a ColdFusion Extension (CFX) procedure.

The CCFXRequest class, CCFXQuery class, and CCFXStringSet class can throw exceptions of this type. Your ColdFusion Extension code must be written to handle exceptions of this type.

Class methods

<table>
<thead>
<tr>
<th>virtual LPCSTR GetError()</th>
<th>The :GetError function returns a general error message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual LPCSTR GetDiagnostics()</td>
<td>The :GetDiagnostics function returns detailed error information.</td>
</tr>
</tbody>
</table>

CCFXException::GetError

*Description*

Provides basic user output for exceptions that occur during processing.

CCFXException::GetDiagnostics

*Description*

Provides detailed user output for exceptions that occur during processing.

*Example*

This code block shows how GetError and GetDiagnostics work with ThrowException and ReThrowException.
// Write output back to the user here...
pRequest->Write( "Hello from CFX_FOO2!" ) ;
pRequest->ThrowException("User Error", "You goof'd...");

// Output optional debug info
if ( pRequest->Debug() )
{
  pRequest->WriteDebug( "Debug info..." ) ;
}

// Catch ColdFusion exceptions & re-raise them
catch( CCFXException* e )
{
  // This is how you would pull the error information
  LPCSTR strError = e->GetError();
  LPCSTR strDiagnostic = e->GetDiagnostics();

  pRequest->ReThrowException( e ) ;
}

// Catch ALL other exceptions and throw them as
// ColdFusion exceptions (DO NOT REMOVE! --
// this prevents the server from crashing in
// case of an unexpected exception)
catch( ... )
{
  pRequest->ThrowException(
      "Error occurred in tag CFX_FOO2",
      "Unexpected error occurred while processing tag." ) ;
}
CCFXQuery class

An abstract class that represents a query used or created by a ColdFusion Extension (CFX). Queries contain one or more columns of data that extend over a varying number of rows.

Class methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual int AddRow()</td>
<td>:AddRow adds a row to a query.</td>
</tr>
<tr>
<td>virtual CCFXStringSet* GetColumns</td>
<td>:GetColumns retrieves a list of a query’s column names.</td>
</tr>
<tr>
<td>virtual LPCSTR GetData( int iRow, int iColumn )</td>
<td>:GetData retrieves a data element from a row and column of a query.</td>
</tr>
<tr>
<td>virtual LPCSTR GetName()</td>
<td>:GetName retrieves the name of a query.</td>
</tr>
<tr>
<td>virtual int GetRowCount()</td>
<td>:GetRowCount retrieves the number of rows in a query.</td>
</tr>
<tr>
<td>virtual void SetData( int iRow, int iColumn, LPCSTR lpszData )</td>
<td>:SetData sets a data element within a row and column of a query.</td>
</tr>
<tr>
<td>virtual void SetQueryString( LPCSTR lpszQuery )</td>
<td>This function is deprecated. It might not work, and might cause an error, in later releases.</td>
</tr>
<tr>
<td>virtual void SetTotalTime( DWORD dwMilliseconds )</td>
<td>This function is deprecated. It might not work, and might cause an error, in later releases.</td>
</tr>
</tbody>
</table>

CCFXQuery::AddRow

Syntax

```
int CCFXQuery::AddRow(void)
```

Description

Add a row to the query. Call this function to append a row to a query.

Returns

Returns the index of the row that was appended to a query.

Example

The following example shows the addition of two rows to a three-column (‘City’, ‘State’, and ‘Zip’) query:
// First row
int iRow;
iRow = pQuery->AddRow();
pQuery->SetData( iRow, iCity, "Minneapolis" ) ;
pQuery->SetData( iRow, iState, "MN" ) ;
pQuery->SetData( iRow, iZip, "55345" ) ;

// Second row
iRow = pQuery->AddRow();
pQuery->SetData( iRow, iCity, "St. Paul" ) ;
pQuery->SetData( iRow, iState, "MN" ) ;
pQuery->SetData( iRow, iZip, "55105" ) ;

CCFXQuery::GetColumns

Syntax

CCFXStringSet* CCFXQuery::GetColumns(void)

Description

Retrieves a list of the column names contained in a query.

Returns

Returns an object of CCFXStringSet that contains a list of the columns in the query. ColdFusion automatically frees the memory that is allocated for the returned string set, after the request is completed.

Example

The following example gets the list of columns, then iterates over the list, writing each column name back to the user:

// Get the list of columns from the query
CCFXStringSet* pColumns = pQuery->GetColumns() ;
int nNumColumns = pColumns->GetCount() ;

// Print the list of columns to the user
pRequest->Write( "Columns in query: " ) ;
for( int i=1; i<=nNumColumns; i++ )
{
  pRequest->Write( pColumns->GetString( i ) ) ;
pRequest->Write( " " ) ;
}

CCFXQuery::GetData

Syntax
LPCSTR CCFXQuery::GetData(int iRow, int iColumn)

**Description**

Gets a data element from a row and column of a query. Row and column indexes begin with 1. You can determine the number of rows in a query by calling `GetRowCount`. You can determine the number of columns in a query by retrieving the list of columns using `GetColumns`, and then calling `GetCount` on the returned string set.

**Returns**

Returns the value of the requested data element.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iRow</td>
<td>Row to retrieve data from (1-based)</td>
</tr>
<tr>
<td>iColumn</td>
<td>Column to retrieve data from (1-based)</td>
</tr>
</tbody>
</table>

**Example**

The following example iterates over the elements of a query and writes the data in the query back to the user in a simple, space-delimited format:

```c
int iRow, iCol;
int nNumCols = pQuery->GetColumns()->GetCount();
int nNumRows = pQuery->GetRowCount();
for (iRow=1; iRow<=nNumRows; iRow++) {
    for (iCol=1; iCol<=nNumCols; iCol++) {
        pRequest->Write( pQuery->GetData( iRow, iCol ) );
        pRequest->Write( " " );
    }
    pRequest->Write( "<BR>" );
}
```

**CCFXQuery::GetName**

**Syntax**

LPCSTR CCFXQuery::GetName(void)

**Description**
Returns the name of a query.

**Example**

The following example retrieves the name of a query and writes it back to the user:

```c
CCFXQuery* pQuery = pRequest->GetQuery() ;
pRequest->Write( "The query name is: " ) ;
pRequest->Write( pQuery->GetName() ) ;
```

**CCFXQuery::GetRowCount**

**Syntax**

```c
int CCFXQuery::GetRowCount(void)
```

**Description**

Returns the number of rows contained in a query.

**Example**

The following example retrieves the number of rows in a query and writes it back to the user:

```c
CCFXQuery* pQuery = pRequest->GetQuery() ;
char buffOutput[256] ;
wsprintf( buffOutput, "The number of rows in the query is %ld.\n", pQuery->GetRowCount() ) ;
pRequest->Write( buffOutput ) ;
```

**CCFXQuery::SetData**

**Syntax**

```c
void CCFXQuery::SetData(int iRow, int iColumn, LPCSTR lpszData)
```

**Description**

Sets a data element within a row and column of a query. Row and column indexes begin with 1. Before calling `SetData` for a given row, call `AddRow` and use the return value as the row index for your call to `SetData`.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
**Example**

The following example shows the addition of two rows to a three-column ('City', 'State', and 'Zip') query:

```java
// First row
int iRow;
iRow = pQuery->AddRow();
pQuery->SetData( iCity, iRow, "Minneapolis" );
pQuery->SetData( iState, iRow, "MN" );
pQuery->SetData( iZip, iRow, "55345" );

// Second row
iRow = pQuery->AddRow();
pQuery->SetData( iCity, iRow, "St. Paul" );
pQuery->SetData( iState, iRow, "MN" );
pQuery->SetData( iZip, iRow, "55105" );
```
CCFXRequest class

Abstract class that represents a request made to a ColdFusion Extension (CFX). An instance of this class is passed to the main function of your extension DLL. The class provides interfaces that can be used by the custom extension for the following actions:

- Reading and writing variables
- Returning output
- Creating and using queries
- Throwing exceptions

Class methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual BOOL AttributeExists( LPCSTR lpszName )</td>
<td>:AttributeExists checks whether the attribute was passed to the tag.</td>
</tr>
<tr>
<td>virtual LPCSTR GetAttribute( LPCSTR lpszName )</td>
<td>:GetAttribute gets the value of the passed attribute.</td>
</tr>
<tr>
<td>virtual CCFXStringSet* GetAttributeList()</td>
<td>:GetAttributeList gets an array of attribute names passed to the tag.</td>
</tr>
<tr>
<td>virtual CCFXQuery* GetQuery()</td>
<td>:GetQuery gets the query that was passed to the tag.</td>
</tr>
<tr>
<td>virtual LPCSTR GetSetting( LPCSTR lpszSettingName )</td>
<td>{{CCFXRequest::GetSetting}} This method is deprecated. It might not work, and might cause an error, in later releases.</td>
</tr>
<tr>
<td>virtual void Write( LPCSTR lpszOutput )</td>
<td>:Write writes text output back to the user.</td>
</tr>
<tr>
<td>virtual void SetVariable( LPCSTR lpszName, LPCSTR lpszValue )</td>
<td>:SetVariable sets a variable in the template that contains this tag.</td>
</tr>
<tr>
<td>virtual CCFXQuery* AddQuery( LPCSTR lpszName, CCFXStringSet* pColumns )</td>
<td>:AddQuery adds a query to the template that contains this tag.</td>
</tr>
<tr>
<td>virtual BOOL Debug()</td>
<td>:Debug checks whether the tag contains the Debug attribute.</td>
</tr>
<tr>
<td>virtual void WriteDebug( LPCSTR lpszOutput )</td>
<td>:WriteDebug writes text output into the debug stream.</td>
</tr>
<tr>
<td>virtual CCFXStringSet* CreateStringSet()</td>
<td>:CreateStringSet allocates and returns a CCFXStringSet instance.</td>
</tr>
<tr>
<td>virtual void ThrowException( LPCSTR lpszError, LPCSTR lpszDiagnostics )</td>
<td>:ThrowException throws an exception and ends processing of this request.</td>
</tr>
</tbody>
</table>
virtual void ReThrowException(CCFXException* e) : ReThrowException rethrows an exception that has been caught.

virtual void SetCustomData(LPVOID lpvData) : SetCustomData sets custom (tag specific) data to carry with a request.

virtual LPVOID GetCustomData() : GetCustomData gets custom (tag specific) data for a request.

CCFXRequest::AddQuery

Syntax

CCFXQuery* CCFXRequest::AddQuery(LPCSTR lpszName, CCFXStringSet* pColumns)

Description

Adds a query to the calling template. The query can be accessed by CFML tags (for example, `cfoutput` or `cftablename`) within the template. After calling `AddQuery`, the query is empty (it has 0 rows). To populate the query with data, call the `AddRow` and `SetData` functions.

Returns

Returns a pointer to the query that was added to the template (an object of class `CCFXQuery`). The memory allocated for the returned query is freed automatically by ColdFusion after the request is completed.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszName</td>
<td>Name of query to add to the template (must be unique)</td>
</tr>
<tr>
<td>pColumns</td>
<td>List of column names to be used in the query</td>
</tr>
</tbody>
</table>

Example

The following example adds a query named ‘People’ to the calling template. The query has two columns (‘FirstName’ and ‘LastName’) and two rows:

```
// Create a string set and add the column names to it
CCFXStringSet* pColumns = pRequest->CreateStringSet();
int iFirstName = pColumns->AddString( "FirstName" ) ;
int iLastName = pColumns->AddString( "LastName" ) ;

// Create a query that contains these columns
CCFXQuery* pQuery = pRequest->AddQuery( "People", pColumns ) ;

// Add data to the query
int iRow ;
iRow = pQuery->AddRow() ;
pQuery->SetData( iRow, iFirstName, "John" ) ;
pQuery->SetData( iRow, iLastName, "Smith" ) ;
iRow = pQuery->AddRow() ;
pQuery->SetData( iRow, iFirstName, "Jane" ) ;
pQuery->SetData( iRow, iLastName, "Doe" ) ;

CCFXRequest::AttributeExists

Syntax

BOOL CCFXRequest::AttributeExists(LPCSTR lpszName)

Description

Checks whether the parameter was passed to the tag. Returns True if the parameter is available; False, otherwise.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszName</td>
<td>Name of the parameter to check (case insensitive)</td>
</tr>
</tbody>
</table>

Example

The following example checks whether the user passed an attribute named DESTINATION to the tag, and throws an exception if the attribute was not passed:

```plaintext
if ( pRequest->AttributeExists("DESTINATION")==FALSE )
{
    pRequest->ThrowException(
        "Missing DESTINATION parameter",
        "You must pass a DESTINATION parameter in "
        "order for this tag to work correctly." ) ;
}
```

CCFXRequest::CreateStringSet
Syntax

CCFXStringSet* CCFXRequest::CreateStringSet(void)

Description

Allocates and returns an instance. Always use this function to create string sets, as opposed to directly using the new operator.

Returns

Returns an object of CCFXStringSet class. The memory allocated for the returned string set is freed automatically by ColdFusion after the request is completed.

Example

The following example creates a string set and adds three strings to it:

```c++
CCFXStringSet* pColors = pRequest->CreateStringSet();
pColors->AddString( "Red" );
pColors->AddString( "Green" );
pColors->AddString( "Blue" );
```

CCFXRequest::Debug

Syntax

BOOL CCFXRequest::Debug(void)

Description

Checks whether the tag contains the Debug attribute. Use this function to determine whether to write debug information for a request. For more information, see :WriteDebug.

Returns

Returns True if the tag contains the Debug attribute; False, otherwise.

Example

The following example checks whether the Debug attribute is present, and if it is, it writes a brief debug message:
```
if ( pRequest->Debug() )
{
    pRequest->WriteDebug( "Top secret debug info" ) ;
}
```
Gets an array of attribute names passed to the tag. To get the value of one attribute, use \texttt{GetAttribute}.

\textbf{Returns}

Returns an object of class \texttt{CCFXStringSet} class that contains a list of attributes passed to the tag. The memory allocated for the returned string set is freed automatically by ColdFusion after the request is completed.

\textbf{Example}

The following example gets the list of attributes and iterates over the list, writing each attribute and its value back to the user.

\begin{verbatim}
LPCSTR lpszName, lpszValue;
CCFXStringSet* pAttribs = pRequest->GetAttributeList();
int nNumAttribs = pAttribs->GetCount();

for( int i=1; i<=nNumAttribs; i++ )
{
    lpszName = pAttribs->GetString( i );
    lpszValue = pRequest->GetAttribute( lpszName );
    pRequest->Write( lpszName );
    pRequest->Write( " = " );
    pRequest->Write( lpszValue );
    pRequest->Write( "<BR>" );
}
\end{verbatim}

\section*{CCFXRequest::GetCustomData}

\textbf{Syntax}

\begin{verbatim}
LPVOID CCFXRequest::GetCustomData(void)
\end{verbatim}

\textbf{Description}

Gets the custom (tag specific) data for the request. This method is typically used from within subroutines of a tag implementation to extract tag data from a request.

\textbf{Returns}

Returns a pointer to the custom data, or \texttt{NULL} if no custom data has been set during this request using \texttt{SetCustomData}.

\textbf{Example}

The following example retrieves a pointer to a request specific data structure of hypothetical type MYTAGDATA:
void DoSomeGruntWork( CCFXRequest* pRequest )
{
    MYTAGDATA* pTagData =
        (MYTAGDATA*)pRequest->GetCustomData() ;
    ...
    ... remainder of procedure ...
}

CCFXRequest::GetQuery

Syntax

CCFXQuery* CCFXRequest::GetQuery(void)

Description

Retrieves a query that was passed to a tag. To pass a query to a custom tag, you use the QUERY attribute. Set the attribute to the name of a query (created using the cfquery tag or another custom tag). The QUERY attribute is optional and must be used only by tags that process an existing data set.

Returns

Returns an object of the CCFXQuery class that represents the query passed to the tag. If no query was passed to the tag, NULL is returned. The memory allocated for the returned query is freed automatically by ColdFusion after the request is completed.

Example

The following example retrieves the query that was passed to the tag. If no query was passed, an exception is thrown:

CCFXQuery* pQuery = pRequest->GetQuery() ;
if ( pQuery == NULL )
{
    pRequest->ThrowException(
        "Missing QUERY parameter",
        "You must pass a QUERY parameter in 
        "order for this tag to work correctly." ) ;
}

CCFXRequest::ReThrowException

Syntax
void CCFXRequest::ReThrowException(CCFXException* e)

**Description**

Rethrows an exception that has been caught within an extension procedure. This function is used to avoid having C++ exceptions that are thrown by DLL extension code propagate back into ColdFusion. Catch ALL C++ exceptions that occur in extension code, and either re-throw them (if they are of the CCFXException class) or create and throw a new exception pointer using :ThrowException.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>A CCFXException that has been caught</td>
</tr>
</tbody>
</table>

**Example**

The following code demonstrates how to handle exceptions in ColdFusion Extension DLL procedures:

```c++
try
{
    ...Code that could throw an exception...
}
catch( CCFXException* e )
{
    ...Do appropriate resource cleanup here...
    // Re-throw the exception
    pRequest->ReThrowException( e ) ;
}
catch( ... )
{
    // Something nasty happened
    pRequest->ThrowException(
        "Unexpected error occurred in CFX tag", "" ) ;
}
```

**CCFXRequest::SetCustomData**

**Syntax**

```c++
void CCFXRequest::SetCustomData(LPVOID lpvData)
```

**Description**

Sets custom (tag specific) data to carry with the request. Use this function to store request specific data to pass to
procedures within your custom tag implementation.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpvData</td>
<td>Pointer to custom data</td>
</tr>
</tbody>
</table>

**Example**

The following example creates a request-specific data structure of hypothetical type MYTAGDATA and stores a pointer to the structure in the request for future use:

```c
void ProcessTagRequest( CCFXRequest* pRequest )
try
{
    MYTAGDATA tagData ;
    pRequest->SetCustomData( (LPVOID)&tagData ) ;
    ... remainder of procedure ...
}
```

**CCFXRequest::SetVariable**

**Syntax**

```c
void CCFXRequest::SetVariable(LPCSTR lpszName, LPCSTR lpszValue)
```

**Description**

Sets a variable in the calling template. If the variable name already exists in the template, its value is replaced. If it does not exist, a variable is created. The values of variables created using `SetVariable` can be accessed in the same manner as other template variables (for example, `#MessageSent#`).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszName</td>
<td>Name of variable</td>
</tr>
<tr>
<td>lpszValue</td>
<td>Value of variable</td>
</tr>
</tbody>
</table>

**Example**

The following example sets the value of a variable named `MessageSent` based on the success of an operation performed by the custom tag:
BOOL bMessageSent;
...attempt to send the message...
if ( bMessageSent == TRUE ) {
    pRequest->SetVariable( "MessageSent", "Yes" );
} else {
    pRequest->SetVariable( "MessageSent", "No" );
}

CCFXRequest::ThrowException

Syntax

void CCFXRequest::ThrowException(LPCSTR lpszError, LPCSTR lpszDiagnostics)

Description

Throws an exception and ends processing of a request. Call this function when you encounter an error that does not allow you to continue processing the request. This function is almost always combined with the :ReThrowExcepti

on to protect against resource leaks in extension code.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszError</td>
<td>Short identifier for error</td>
</tr>
<tr>
<td>lpszDiagnostics</td>
<td>Error diagnostic information</td>
</tr>
</tbody>
</table>

Example

The following example throws an exception indicating that an unexpected error occurred while processing a request:

```c
char buffError[512] ;
wsprintf( buffError,
    "Unexpected Windows NT error number %ld 
    "occurred while processing request.", GetLast Error() ) ;

pRequest->ThrowException( "Error occurred", buffError ) ;
```
void CCFXRequest::Write LPCSTR lpszOutput

Description
Writes text output back to the user.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszOutput</td>
<td>Text to output</td>
</tr>
</tbody>
</table>

Example
The following example creates a buffer to hold an output string, fills the buffer with data, and writes the output back to the user:

```c
CHAR buffOutput[1024] ;
wsprintf( buffOutput, "The destination is: %s", pRequest->GetAttribute("DESTINATION") ) ;
pRequest->Write( buffOutput ) ;
```

CCFXRequest::WriteDebug

Syntax

void CCFXRequest::WriteDebug LPCSTR lpszOutput

Description
Writes text output into the debug stream. The text is only displayed to the end user if the tag contains the Debug attribute. (For more information, see `:Debug`.)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszOutput</td>
<td>Text to output</td>
</tr>
</tbody>
</table>

Example
The following example checks whether the Debug attribute is present; if so, it writes a brief debug message:
if ( pRequest->Debug() )
{
    pRequest->WriteDebug( "Top secret debug info" ) ;
}
**CCFXStringSet class**

Abstract class that represents a set of ordered strings. You can add strings to a set and retrieve them by a numeric index (index values for strings are 1-based). To create a string set, use `CreateStringSet`.

**Class methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual int AddString( LPCSTR lpszString)</td>
<td><code>AddString</code> adds a string to the end of a list.</td>
</tr>
<tr>
<td>virtual int GetCount()</td>
<td><code>GetCount</code> gets the number of strings contained in a list.</td>
</tr>
<tr>
<td>virtual LPCSTR GetString( int iIndex )</td>
<td><code>GetString</code> gets the string located at the passed index.</td>
</tr>
<tr>
<td>virtual int GetIndexForString( LPCSTR lpszString)</td>
<td><code>GetIndexForString</code> gets the index for the passed string.</td>
</tr>
</tbody>
</table>

**CCFXStringSet::AddString**

**Syntax**

```cpp
int CCFXStringSet::AddString(LPCSTR lpszString)
```

**Description**

Adds a string to the end of the list.

**Returns**

The index of the string that was added.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpszString</td>
<td>String to add to the list</td>
</tr>
</tbody>
</table>

**Example**

The following example demonstrates adding three strings to a string set and saving the indexes of the items that are added:

```cpp
CCFXStringSet* pSet = pRequest->CreateStringSet() ;
int iRed = pSet->AddString( "Red" ) ;
int iGreen = pSet->AddString( "Green" ) ;
int iBlue = pSet->AddString( "Blue" ) ;
```
CCFXStringSet::GetCount

Syntax

```c
int CCFXStringSet::GetCount(void)
```

Description

Gets the number of strings in a string set. The value can be used with :GetString to iterate over the strings in the set (recall that the index values for strings in the list begin at 1).

Returns

Returns the number of strings contained in the string set.

Example

The following example demonstrates using GetCount with :GetString to iterate over a string set and write the contents of the list back to the user:

```c
int nNumItems = pStringSet->GetCount() ;
for ( int i=1; i<=nNumItems; i++ )
{
  pRequest->Write( pStringSet->GetString( i ) ) ;
pRequest->Write( "<BR>" ) ;
}
```

CCFXStringSet::GetIndexForString

Syntax

```c
int CCFXStringSet::GetIndexForString(LPCSTR lpszString)
```

Description

Searches for a passed string. The search is case-insensitive.

Returns

If the string is found, its index within the string set is returned. If it is not found, the constant CFX_STRING_NOT_FOUND is returned.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
lpszString | String to search for

**Example**

The following example demonstrates a search for a string and throwing an exception if it is not found:

```cpp
CCFXStringSet* pAttribs = pRequest->GetAttributeList() ;
int iDestination = pAttribs->GetIndexForString("DESTINATION") ;
if ( iDestination == CFX_STRING_NOT_FOUND )
{
    pRequest->ThrowException(
        "DESTINATION attribute not found."
        "The DESTINATION attribute is required "
        "by this tag." ) ;
}
```

**CCFXStringSet::GetString**

**Syntax**

```cpp
LPCSTR CCFXStringSet::GetString(int iIndex)
```

**Description**

Retrieves the string located at the passed index (index values are 1-based).

**Returns**

Returns the string located at the passed index.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iIndex</td>
<td>Index of string to retrieve</td>
</tr>
</tbody>
</table>

**Example**

The following example demonstrates `GetString` with `::GetCount` to iterate over a string set and write the contents of a list back to the user:
```csharp
int nNumItems = pStringSet->GetCount() ;
for ( int i=1; i<=nNumItems; i++ )
{
   pRequest->Write( pStringSet->GetString( i ) ) ;
   pRequest->Write( "<BR>" ) ;
}
```

- **Class methods**
  - `CCFXStringSet::AddString`
    - **Syntax**
    - **Description**
    - **Returns**
    - **Parameters**
    - **Example**
  
  - `CCFXStringSet::GetCount`
    - **Syntax**
    - **Description**
    - **Returns**
    - **Parameters**
    - **Example**
  
  - `CCFXStringSet::GetIndexForString`
    - **Syntax**
    - **Description**
    - **Returns**
    - **Parameters**
    - **Example**
  
  - `CCFXStringSet::GetString`
    - **Syntax**
    - **Description**
    - **Returns**
    - **Parameters**
    - **Example**
ColdFusion Java CFX Reference

ColdFusion includes Java interfaces for building ColdFusion custom CFXs in Java.

- Class libraries overview
- Custom tag interface
- Query interface
- Request interface
- Response interface
- Debugging classes reference
## Class libraries overview

The following Java interfaces are available for building ColdFusion custom CFXs in Java:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Custom tag interface</strong></td>
<td>processRequest</td>
</tr>
<tr>
<td><strong>Query interface</strong></td>
<td>addRow, getColumnIndex, getColumns, getData, getName, getRowCount, setData</td>
</tr>
<tr>
<td><strong>Request interface</strong></td>
<td>attributeExists, debug, getAttribute, getAttributeList, getIntAttribute, getQuery, getSetting</td>
</tr>
<tr>
<td><strong>Response interface</strong></td>
<td>addQuery, SetVariable, write, writeDebug</td>
</tr>
</tbody>
</table>

[back to top]
Custom tag interface

```
public abstract interface CustomTag
```

Interface for implementing custom tags. Classes that implement this interface can be specified in the `CLASS` attribute of the Java CFX tag. For example, in a class `MyCustomTag`, which implements this interface, the following CFML code calls the `MyCustomTag.processRequest` method:

```
<CFX_MyCustomTag>
```

Other attributes can be passed to the Java CFX tag. Their values are available using the Request object passed to the `processRequest` method.

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>processRequest</code></td>
<td><code>public void processRequest(Request request, Response response)</code></td>
<td>Processes a request originating from the <code>CFX_mycustomtag</code> tag</td>
</tr>
</tbody>
</table>

```
processRequest
```

Description

Processes a request originating from the Java CFX tag.

Category

Custom tag interface

Syntax

```
public void processRequest(Request request, Response response)
```

Throws

- `Exception` If an unexpected error occurs while processing the request.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>Parameters (attributes, query, and so on.) for this request</td>
</tr>
<tr>
<td>response</td>
<td>Interface for generating response to request (output, variables, queries, and so on)</td>
</tr>
</tbody>
</table>

**back to top**
Query interface

public abstract interface Query

Interface to a query used or created by a custom tag. A query contains tabular data organized by named columns and rows.

Methods

<table>
<thead>
<tr>
<th>Returns</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>addRow()</td>
<td>Adds a row to the query</td>
</tr>
<tr>
<td>int</td>
<td>getColumnIndex(String name)</td>
<td>Gets the index of a column given its name</td>
</tr>
<tr>
<td>String[]</td>
<td>getColumns()</td>
<td>Gets a list of the column names in a query</td>
</tr>
<tr>
<td>String</td>
<td>getData(int iRow, int iCol)</td>
<td>Gets a data element from a row and column of a query</td>
</tr>
<tr>
<td>String</td>
<td>getName()</td>
<td>Gets the name of a query</td>
</tr>
<tr>
<td>int</td>
<td>getRowCount()</td>
<td>Gets the number of rows in a query</td>
</tr>
<tr>
<td>void</td>
<td>setData(int iRow, int iCol, String data)</td>
<td>Sets a data element in a row and column of a query</td>
</tr>
</tbody>
</table>

addRow

**Description**

Adds a row to a query. Call this method to append a row to a query. Returns the index of the row that was appended to the query.

**Category**

Query interface

**Syntax**

```java
public int addRow()
```

See also
**setData, getData**

**Example**

The following example demonstrates the addition of two rows to a query that has three columns, *City, State,* and *Zip*:

```java
// Define column indexesint iCity = 1, iState = 2, iZip = 3;

// First row
int iRow = query.addRow();
query.setData(iRow, iCity, "Minneapolis");
query.setData(iRow, iState, "MN");
query.setData(iRow, iZip, "55345");

// Second row
iRow = query.addRow();
query.setData(iRow, iCity, "St. Paul");
query.setData(iRow, iState, "MN");
query.setData(iRow, iZip, "55105");
```

**getColumnIndex**

**Description**

Returns the index of the column, or 0 if no such column exists.

**Category**

Query interface

**Syntax**

```java
public int getColumnIndex(String name)
```

**See also**

`getColumns, getData`

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of column to get index of (lookup is case-insensitive)</td>
</tr>
</tbody>
</table>

**Example**

The following example retrieves the index of the EMAIL column and uses it to output a list of the addresses contained in the column:
// Get the index of the EMAIL column
int iEMail = query.getColumnIndex( "EMAIL" ) ;

// Iterate over the query and output list of addresses
int nRows = query.getRowCount() ;
for( int iRow = 1; iRow <= nRows; iRow++ )
{
    response.write( query.getData( iRow, iEMail ) + "<BR>" ) ;
}

getColumns

Description

Returns an array of strings containing the names of the columns in the query.

Category

Query interface

Syntax

public String[] getColumns()

Example

The following example retrieves the array of columns, then iterates over the list, writing each column name back to the user:

// Get the list of columns from the query
String[] columns = query.getColumns() ;
int nNumColumns = columns.length ;

// Print the list of columns to the user
response.write( "Columns in query: " ) ;
for( int i=0; i<nNumColumns; i++ )
{
    response.write( columns[i] + " " ) ;
}

data

Description

Retrieves a data element from a row and column of a query. Row and column indexes begin with 1. You can find the number of rows in a query by calling getRowCount. You can find the number of columns in a query by calling getColumns. Returns the value of the requested data element.
**Category**

**Query interface**

**Syntax**

```java
public String getData(int iRow, int iCol)
```

**Throws**

`IndexOutOfBoundsException` if an invalid index is passed to the method.

**See also**

`setData`, `addRow`

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iRow</td>
<td>Row to retrieve data from (1-based)</td>
</tr>
<tr>
<td>iCol</td>
<td>Column to retrieve data from (1-based)</td>
</tr>
</tbody>
</table>

**Example**

The following example iterates over the rows of a query and writes the data back to the user in a simple, space-delimited format:

```java
int iRow, iCol;
int nNumCols = query.getColumns().length;
int nNumRows = query.getRowCount();
for (iRow = 1; iRow <= nNumRows; iRow++) {
  for (iCol = 1; iCol <= nNumCols; iCol++) {
    response.write(query.getData(iRow, iCol) + " ");
  }
  response.write("<BR>");
}
```

**getName**

**Description**

Returns the name of a query.

**Category**

**Query interface**
**Syntax**

```java
public String getName()
```

**Example**

The following example retrieves the name of a query and writes it back to the user:

```java
Query query = request.getQuery() ;
response.write( "The query name is: " + query.getName() ) ;
```

**getRowCount**

**Description**

Retrieves the number of rows in a query. Returns the number of rows contained in a query.

**Category**

*Query interface*

**Syntax**

```java
public int getRowCount()
```

**Example**

The following example retrieves the number of rows in a query and writes it back to the user:

```java
Query query = request.getQuery() ;
int rows = query.getRowCount() ;
response.write( "The number of rows in the query is " + Integer.ToString(rows) ) ;
```

**setData**

**Description**

Sets a data element in a row and column of a query. Row and column indexes begin with 1. Before calling `setData` for a given row, call `addRow` and use the return value as the row index for your call to `setData`.

**Category**

*Query interface*

**Syntax**
public void setData(int iRow, int iCol, String data)

Throws

IndexOutOfBoundsException if an invalid index is passed to the method.

See also

gedata, addRow

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iRow</td>
<td>Row of data element to set (1-based)</td>
</tr>
<tr>
<td>iCol</td>
<td>Column of data element to set (1-based)</td>
</tr>
<tr>
<td>data</td>
<td>New value for data element</td>
</tr>
</tbody>
</table>

Example

The following example demonstrates the addition of two rows to a query that has three columns, City, State, and Zip:

```java
// Define column indexes
int iCity = 1, iState = 2, iZip = 3;

// First row
int iRow = query.addRow();
query.setData(iRow, iCity, "Minneapolis");
query.setData(iRow, iState, "MN");
query.setData(iRow, iZip, "55345");

// Second row
iRow = query.addRow();
query.setData(iRow, iCity, "St. Paul");
query.setData(iRow, iState, "MN");
query.setData(iRow, iZip, "55105");
```
Request interface

```java
public abstract interface Request
```

Interface to a request made to a CustomTag. The interface includes methods for retrieving attributes passed to the tag (including queries) and reading global tag settings.

### Methods

<table>
<thead>
<tr>
<th>Returns</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td><code>attributeExists(String name)</code></td>
<td>Checks whether the attribute was passed to this tag.</td>
</tr>
<tr>
<td>boolean</td>
<td><code>debug()</code></td>
<td>Checks whether the tag contains the <code>debug</code> attribute.</td>
</tr>
<tr>
<td>String</td>
<td><code>getAttribute(String name)</code></td>
<td>Retrieves the value of the passed attribute.</td>
</tr>
<tr>
<td>String[]</td>
<td><code>getAttributeList()</code></td>
<td>Retrieves a list of attributes passed to the tag.</td>
</tr>
<tr>
<td>int</td>
<td><code>getIntAttribute(String name)</code></td>
<td>Retrieves the value of the passed attribute as an integer.</td>
</tr>
<tr>
<td>int</td>
<td><code>getIntAttribute(String name, int def)</code></td>
<td>Retrieves the value of the passed attribute as an integer (returns default if the attribute does not exist or is not a valid number).</td>
</tr>
<tr>
<td>Query</td>
<td><code>getQuery()</code></td>
<td>Retrieves the query that was passed to this tag.</td>
</tr>
</tbody>
</table>

### `attributeExists`

**Description**

Checks whether the attribute was passed to this tag. Returns True if the attribute is available; otherwise returns False.

**Category**

Request interface

**Syntax**
public boolean attributeExists(String name)

See also
getAttribute, getAttributeList

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the attribute to check (case-insensitive)</td>
</tr>
</tbody>
</table>

Example

The following example checks whether the user passed an attribute named DESTINATION to the tag; if not, it throws an exception:

```java
if ( ! request.attributeExists("DESTINATION") )
{
    throw new Exception(
        "Missing DESTINATION parameter",
        "You must pass a DESTINATION parameter in "
        "order for this tag to work correctly." ) ;
}
```

debug

Description

Checks whether the tag contains the debug attribute. Use this method to determine whether to write debug information for this request. For more information, see writeDebug. Returns True if the tag contains the debug attribute; False, otherwise.

Category

Request interface

Syntax

```java
public boolean debug()
```

See also

writeDebug

Example
The following example checks whether the `debug` attribute is present, and if so, it writes a brief debug message:

```java
if ( request.debug() )
{
    response.writeDebug( "debug info" ) ;
}
```

### getAttribute

**Description**

Retrieves the value of a passed attribute. Returns an empty string if the attribute does not exist (use `attributeExists` to test whether an attribute was passed to the tag). Use `getAttribute(String, String)` to return a default value rather than an empty string. Returns the value of the attribute passed to the tag. If no attribute of that name was passed to the tag, an empty string is returned.

**Category**

Request interface

**Syntax**

```java
public String getAttribute(String name)
```

**See also**

`attributeExists`, `getAttributeList`, `getIntAttribute`

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The attribute to retrieve (case-insensitive)</td>
</tr>
</tbody>
</table>

**Example**

The following example retrieves an attribute named `DESTINATION` and writes its value back to the user:

```java
String strDestination = request.getAttribute("DESTINATION") ;
response.write( "The destination is: " + strDestination ) ;
```

### getAttributeList

**Description**

Retrieves a list of attributes passed to the tag. To retrieve the value of one attribute, use the `getAttribute` method.
d. Returns an array of strings containing the names of the attributes passed to the tag.

**Category**

**Request interface**

**Syntax**

```
public String[] getAttributeList()
```

**See also**

attributeExists

**Example**

The following example retrieves the list of attributes, then iterates over the list, writing each attribute and its value back to the user:

```
String[] attribs = request.getAttributeList() ;
int nNumAttribs = attribs.length ;
for( int i = 0; i < nNumAttribs; i++ )
{
    String strName = attribs[i] ;
    String strValue = request.getAttribute( strName ) ;
    response.write( strName + "=" + strValue + "<BR>" ) ;
}
```

ggetIntAttribute

**Description**

Retrieves the value of the passed attribute as an integer. Returns -1 if the attribute does not exist. Use attributeExists to test whether an attribute was passed to the tag. Use getIntAttribute(String, int) to return a default value rather than throwing an exception or returning -1. Returns the value of the attribute passed to the tag. If no attribute of that name was passed to the tag, -1 is returned.

**Category**

**Request interface**

**Syntax**

```
public int getIntAttribute(String name)
```

**Throws**
NumberFormatException if the attribute is not a valid number.

See also

attributeExists, getAttributeList

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The attribute to retrieve (case-insensitive)</td>
</tr>
</tbody>
</table>

Example

The following example retrieves an attribute named PORT and writes its value back to the user:

```java
int nPort = request.getIntAttribute("PORT") ;
if ( nPort != -1 )
    response.write( "The port is: " + String.valueOf(nPort) ) ;
```

ggetQuery

Description

Retrieves the query that was passed to this tag. To pass a query to a custom tag, you use the query attribute. It should be set to the name of a query (created using the cfquery tag). The query attribute is optional and should be used only by tags that process an existing dataset. Returns the Query that was passed to the tag. If no query was passed, returns null.

Category

Request interface

Syntax

```java
public Query getQuery()
```

Example

The following example retrieves a query that was passed to a tag. If no query was passed, an exception is thrown:
Query query = request.getQuery() ;
if ( query == null )
{
    throw new Exception(
    "Missing QUERY parameter. " +
    "You must pass a QUERY parameter in "
    "order for this tag to work correctly." ) ;
}

getSetting

Description

Retrieves the value of a global custom tag setting. Custom tag settings are stored in the CustomTags section of the ColdFusion Registry key. Returns the value of the custom tag setting. If no setting of that name exists, an empty string is returned.

Category

Request interface

Syntax

public String getSetting(String name)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the setting to retrieve (case-insensitive)</td>
</tr>
</tbody>
</table>

Usage

All custom tags implemented in Java share a registry key for storing settings. To avoid name conflicts, preface the names of settings with the name of your custom tag class. For example, the code below retrieves the value of a setting named VerifyAddress for a custom tag class named MyCustomTag:

```java
String strVerify = request.getSetting("MyCustomTag.VerifyAddress") ;
if ( Boolean.valueOf(strVerify) )
{
    // Do address verification...
}
```
## Response interface

```java
public abstract interface Response
```

Interface to response generated from a custom tag. This interface includes methods for writing output, generating queries, and setting variables in the calling page.

### Methods

<table>
<thead>
<tr>
<th>Returns</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td><code>addQuery(String name, String[] columns)</code></td>
<td>Adds a query to the calling template.</td>
</tr>
<tr>
<td>void</td>
<td><code>SetVariable(String name{{, String}} value)</code></td>
<td>Sets a variable in the calling template.</td>
</tr>
<tr>
<td>void</td>
<td><code>write(String output)</code></td>
<td>Outputs text back to the user.</td>
</tr>
<tr>
<td>void</td>
<td><code>writeDebug(String output)</code></td>
<td>Writes text output into the debug stream.</td>
</tr>
</tbody>
</table>

**addQuery**

**Description**

Adds a query to the calling template. The query can be accessed by CFML tags in the template. After calling `addQuery`, the query is empty (it has 0 rows). To populate the query with data, call the Query methods `addRow` and `setData`. Returns the Query that was added to the template.

**Category**

Response interface

**Syntax**

```java
public Query addQuery(String name, String[] columns)
```

**Throws**

- `IllegalArgumentException` If the name parameter is not a valid CFML variable name.

**See also**

addRow, `setData`

**Parameters**
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the query to add to the template</td>
</tr>
<tr>
<td>columns</td>
<td>The column names to use in the query</td>
</tr>
</tbody>
</table>

**Example**

The following example adds a query named *People* to the calling template. The query has two columns (*FirstName* and *LastName*) and two rows:

```java
// Create string array with column names (also track columns indexes)
String[] columns = { "FirstName", "LastName" };
int iFirstName = 1, iLastName = 2;

// Create a query which contains these columns
Query query = response.addQuery( "People", columns ) ;

// Add data to the query
int iRow = query.addRow() ;
query.setData( iRow, iFirstName, "John" ) ;
query.setData( iRow, iLastName, "Smith" ) ;
iRow = query.addRow() ;
query.setData( iRow, iFirstName, "Jane" ) ;
query.setData( iRow, iLastName, "Doe" ) ;
```

**setVariable**

**Description**

Sets a variable in the calling template. If the variable name specified exists in the template, its value is replaced. If it does not exist, a new variable is created.

**Category**

Response interface

**Syntax**

```java
public void setVariable(String name, String value)
```

**Throws**

- *IllegalArgumentException* If the *name* parameter is not a valid CFML variable name.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>The name of the variable to set</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>value</td>
<td>The value to set the variable to</td>
</tr>
</tbody>
</table>

**Example**

For example, this code sets the value of a variable named `MessageSent` based on the success of an operation performed by the custom tag:

```java
boolean bMessageSent;

...attempt to send the message...

if ( bMessageSent == true )
{
    response.setVariable( "MessageSent", "Yes" ) ;
}
else
{
    response.setVariable( "MessageSent", "No" ) ;
}
```

**write**

**Description**

Outputs text back to the user.

**Category**

**Response interface**

**Syntax**

```java
public void write(String output)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>Text to output</td>
</tr>
</tbody>
</table>

**Example**

The following example outputs the value of the `DESTINATION` attribute:
writeDebug

Description

Writes text output into the debug stream. This text is displayed to the end-user only if the tag contains the `debug` attribute (check for this attribute using the `Request.debug` method).

Category

Response interface

Syntax

```java
public void writeDebug(String output)
```

See also

depend

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>The text to output</td>
</tr>
</tbody>
</table>

Example

The following example checks whether the `debug` attribute is present; if so, it writes a brief debug message:

```java
if ( request.debug() )
{
    response.writeDebug( "debug info" ) ;
}
```

#back to top
Debugging classes reference

The constructors and methods supported by the DebugRequest, DebugResponse, and DebugQuery classes are as follows. These classes also support the other methods of the Request, Response, and Query interfaces, respectively.

DebugRequest

```java
// initialize a debug request with attributes
public DebugRequest( Hashtable attributes ) ;

// initialize a debug request with attributes and a query
public DebugRequest( Hashtable attributes, Query query ) ;

// initialize a debug request with attributes, a query, and settings
public DebugRequest( Hashtable attributes, Query query, Hashtable settings ) ;
```

DebugResponse

```java
// initialize a debug response
public DebugResponse() ;

// print the results of processing
public void printResults() ;
```

DebugQuery

```java
// initialize a query with name and columns
public DebugQuery( String name, String[] columns ) throws IllegalArgumentException ;

// initialize a query with name, columns, and data
public DebugQuery( String name, String[] columns, String[][] data ) throws IllegalArgumentException ;
```

- [DebugRequest](#)
- [DebugResponse](#)
- [DebugQuery](#)
WDDX JavaScript Objects

You use JavaScript objects and functions to use with WDDX in a ColdFusion application.

- [JavaScript object overview](#)
- [WddxRecordset object](#)
- [WddxSerializer object](#)
JavaScript object overview

These are the JavaScript objects and functions:

<table>
<thead>
<tr>
<th>Class</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>WddxSerializer object</td>
<td>serialize</td>
</tr>
<tr>
<td></td>
<td>serializeVariable</td>
</tr>
<tr>
<td></td>
<td>serializeValue</td>
</tr>
<tr>
<td></td>
<td>write</td>
</tr>
<tr>
<td>WddxRecordset object</td>
<td>addColumn</td>
</tr>
<tr>
<td></td>
<td>addRows</td>
</tr>
<tr>
<td></td>
<td>getField</td>
</tr>
<tr>
<td></td>
<td>getRowCount</td>
</tr>
<tr>
<td></td>
<td>setField</td>
</tr>
<tr>
<td></td>
<td>wddxSerialize</td>
</tr>
</tbody>
</table>

WDDX JavaScript objects are defined in the wddx.js file; this file is installed in the CFIDE/scripts directory. To use these objects, you must put a JavaScript tag before the code that refers to the objects; for example:

```html
<script type="text/javascript" src="/CFIDE/scripts/wddx.js"></script>
```

#back to top
WddxRecordset object

Includes functions that you call as needed when constructing a WDDX record set. For more information on using this object, see Using WDDX in the Developing ColdFusion Applications.

Functions

<table>
<thead>
<tr>
<th>Function syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object.&lt;code&gt;addColumn&lt;/code&gt; (name)</td>
<td>Adds a column to all rows in a WddxRecordset instance.</td>
</tr>
<tr>
<td>object.&lt;code&gt;addRows&lt;/code&gt; (n)</td>
<td>Adds rows to all columns in a WddxRecordset instance.</td>
</tr>
<tr>
<td>object.&lt;code&gt;dump&lt;/code&gt; (escapeStrings)</td>
<td>Displays WddxRecordset object data.</td>
</tr>
<tr>
<td>object.&lt;code&gt;getField&lt;/code&gt; (row, col)</td>
<td>Returns the element in a row/column position.</td>
</tr>
<tr>
<td>object.&lt;code&gt;getRowCount&lt;/code&gt; ()</td>
<td>Indicates the number of rows in a WddxRecordset instance.</td>
</tr>
<tr>
<td>object.&lt;code&gt;setField&lt;/code&gt; (row, col, value)</td>
<td>Sets the element in a row/column position.</td>
</tr>
<tr>
<td>object.&lt;code&gt;wddxSerialize&lt;/code&gt; (serializer)</td>
<td>Serializes a record set.</td>
</tr>
</tbody>
</table>

Returns

HTML table of the WddxRecordset object data.

Usage

Convenient for debugging and testing record sets. The boolean parameter `escapeStrings` determines whether `<` & `>` characters in string values are escaped as `&lt;` & `&gt;` in HTML.

Example
### addColumn

**Description**

Adds a column to all rows in a WddxRecordset instance.

**Syntax**

```javascript
object.addColumn(name)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of the WddxRecordset object</td>
</tr>
<tr>
<td>name</td>
<td>Name of the column to add</td>
</tr>
</tbody>
</table>

**Return value**

None.

**Usage**

Adds a column to every row of the WDDX record set. Initially the new column’s values are set to NULL.

**Example**

This example calls the `addColumn` function:
// Create a new record set
rs = new WddxRecordset();

// Add a new column
rs.addColumn("NewColumn");

// Extend the record set by 3 rows
rs.addRows(3);

// Set an element in the first row
// newValue is a previously defined variable
rs.setField(0, "NewColumn", newValue);

## addRows

**Description**

Adds rows to all columns in a WddxRecordset instance.

**Syntax**

```
object.addRows( n )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of the WddxRecordset object</td>
</tr>
<tr>
<td>n</td>
<td>Integer; number of rows to add</td>
</tr>
</tbody>
</table>

**Return value**

None.

**Usage**

This function adds the specified number of rows to every column of a WDDX record set. Initially, the row/column values are set to NULL.

**Example**

This example calls the `addRows` function:
// Create a new record set
rs = new WddxRecordset();

// Add a new column
rs.addColumn("NewColumn");

// Extend the record set by 3 rows
rs.addRows(3);

// Set an element in the first row
// newValue is a previously defined variable
rs.setField(0, "NewColumn", newValue);

generic

**Description**

Returns the element in the specified row/column position.

**Syntax**

```javascript
object.getField( row, col )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of the WddxRecordset object</td>
</tr>
<tr>
<td>row</td>
<td>Integer; zero-based row number of the value to return</td>
</tr>
<tr>
<td>col</td>
<td>Integer or string; column of the value to be returned.</td>
</tr>
</tbody>
</table>

**Return value**

Returns the value in the specified row/column position.

**Usage**

Call this function to access a value in a WDDX record set.

**Example**

This example calls the `getField` function (the variable `r` is a reference to a WddxRecordset instance):
for (row = 0; row < nRows; ++row)
{
    o += "<tr>";
    for (i = 0; i < colNames.length; ++i)
    {
        o += "<td>" + r.getField(row, colNames[i]) + "</td>";
    }
    o += "</tr>";
}

getRowCount

**Description**

Indicates the number of rows in a WddxRecordset instance.

**Syntax**

```
object.getRowCount()
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of a WddxRecordset object</td>
</tr>
</tbody>
</table>

**Return value**

Integer. Returns the number of rows in the WddxRecordset instance.

**Usage**

Call this function before a looping construct to determine the number of rows in a record set.

**Example**

This example calls the `getRowCount` function:
function dumpWddxRecordset(r)
{
    // Get row count
    nRows = r.getRowCount();
    ...
    for (row = 0; row < nRows; ++row)
    ...
}

setField

Description
Sets the element in the specified row/column position.

Syntax

object.setField( row, col, value )

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of a WddxRecordset object</td>
</tr>
<tr>
<td>row</td>
<td>Integer; row that contains the element to set</td>
</tr>
<tr>
<td>col</td>
<td>Integer or string; the column containing the element to set</td>
</tr>
<tr>
<td>value</td>
<td>Value to set</td>
</tr>
</tbody>
</table>

Return value

None.

Usage

Call this function to set a value in a WddxRecordset instance.

Example

This example calls the `setField` function:
```javascript
// Create a new recordsets = new WddxRecordset();

// Add a new column
rs.addColumn("NewColumn");

// Extend the record set by 3 rows
rs.addRows(3);

// Set an element in the first row
// newValue is a previously defined variable
rs.setField(0, "NewColumn", newValue);
```

## wddxSerialize

### Description

Serializes a record set.

### Syntax

```javascript
object.wddxSerialize( serializer )
```

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of the WddxRecordset object</td>
</tr>
<tr>
<td>serializer</td>
<td>WddxSerializer instance</td>
</tr>
</tbody>
</table>

### Return value

Returns a Boolean True if serialization was successful; False, otherwise.

### Usage

This is an internal function; you do not typically call it.

### Example

This example is from the WddxSerializer `serializeValue` function:
... else if (typeof(obj) == "object") {
    if (obj == null) {
        // Null values become empty strings
        this.write("<string></string>");
    } else if (typeof(obj.wddxSerialize) == "function") {
        // Object knows how to serialize itself
        bSuccess = obj.wddxSerialize(this);
    }
...
WddxSerializer object

The WddxSerializer object includes functions that serialize any JavaScript data structure. For more information on using this object, see Using WDDX in the Developing ColdFusion Applications.

Functions

The only function that developers typically call is serialize.

<table>
<thead>
<tr>
<th>Function syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object.serialize(rootobj)</code></td>
<td>Creates a WDDX packet for a passed WddxRecordset instance.</td>
</tr>
<tr>
<td><code>object.serializeVariable(name, obj)</code></td>
<td>Serializes a property of a structure. If an object is not a string, number, array, Boolean, or a date, WddxSerializer treats it as a structure.</td>
</tr>
<tr>
<td><code>object.serializeValue(obj)</code></td>
<td>Recursively serializes eligible data in a passed instance.</td>
</tr>
<tr>
<td><code>object.write(str)</code></td>
<td>Appends data to the serialized data stream.</td>
</tr>
</tbody>
</table>

serialize

**Description**

Creates a WDDX packet for a passed WddxRecordset instance.

**Syntax**

```
object.serialize( rootobj )
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of the WddxSerializer object</td>
</tr>
<tr>
<td>rootobj</td>
<td>JavaScript data structure to serialize</td>
</tr>
</tbody>
</table>

**Return value**

Returns a serialized WDDX packet as a string if the function succeeds, or a null value if an error occurs.

**Usage**

Call this function to serialize the data in a WddxRecordset instance.
Example

This example shows a JavaScript function that you can call to serialize a WddxRecordset instance. It copies serialized data to a form field for display:

```javascript
function serialize(data, formField)
{
    wddxSerializer = new WddxSerializer();
    wddxPacket = wddxSerializer.serialize(data);
    if (wddxPacket != null)
    {
        formField.value = wddxPacket;
    }
    else
    {
        alert("Couldn't serialize data");
    }
}
```

serializeVariable

Description

Serializes a property of a structure. If an object is not a string, number, array, Boolean, or date, WddxSerializer treats it as a structure.

Syntax

```javascript
object.serializeVariable( name, obj )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of a WddxSerializer object</td>
</tr>
<tr>
<td>name</td>
<td>Property to serialize</td>
</tr>
<tr>
<td>obj</td>
<td>Instance name of the value to serialize</td>
</tr>
</tbody>
</table>

Return value

Returns a Boolean True if serialization was successful; False, otherwise. This is an internal function; you do not typically call it.

Example

This example is from the WddxSerializer `serializeValue` function:
...  
// Some generic object; treat it as a structure  
this.write("<struct>");
for (prop in obj)
{
    bSuccess = this.serializeVariable(prop, obj[prop]);
    if (! bSuccess)
    {
        break;
    }
}
this.write("</struct>");
...

serializeValue

Description
Recursively serializes eligible data in a passed instance. Eligible data includes:

- String
- Number
- Boolean
- Date
- Array
- Recordset
- Any JavaScript object
This function serializes null values as empty strings.

Syntax

object.serializeValue( obj )

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of the WddxSerializer object</td>
</tr>
<tr>
<td>obj</td>
<td>Instance name of the WddxRecordset object to serialize</td>
</tr>
</tbody>
</table>

Return value

Returns a Boolean True if obj was serialized successfully; False, otherwise.

Usage

This is an internal function; you do not typically call it.
Example

This example is from the WddxSerializer serialize function:

```javascript
... 
this.wddxPacket = "";
this.write("<wddxPacket version='1.0'><header/><data>");
bSuccess = this.serializeValue(rootObj);
this.write("</data></wddxPacket>");
if (bSuccess)
{
  return this.wddxPacket;
}
else
{
  return null;
}
... 
```

write

Description

Appends data to a serialized data stream.

Syntax

```javascript
object.write( str )
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Instance name of the WddxSerializer object</td>
</tr>
<tr>
<td>str</td>
<td>String to be copied to the serialized data stream</td>
</tr>
</tbody>
</table>

Return value

Returns an updated serialized data stream as a String.

Usage

This is an internal function; you do not typically call it.

Example

This example is from the WddxSerializer serializeValue function:
else if (typeof(obj) == "number")
{
    // Number value
    this.write("<number>" + obj + "</number>");
}
else if (typeof(obj) == "boolean")
{
    // Boolean value
    this.write("<boolean value='" + obj + "'/>");
}
ColdFusion ActionScript Functions

ColdFusion includes two server-side ActionScript functions, `CF.query` and `CF.http`, including specific syntax and methods.

- `CF.http`
- `CF.query`
CF.http

Description

Executes HTTP POST and GET operations on files. (POST operations upload MIME file types to a server, or post cookie, formfield, URL, file, or CGI variables directly to a server.)

Return value

Returns an object containing properties that you reference to access data.

Syntax

```cfhttp
{method: "get or post",
url: "URL",
username: "username",
password: "password",
resolveurl: "yes or no",
params: arrayvar,
path: "path",
file: "filename"
}
```

Arguments

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Req/Opt</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>Required</td>
<td>One of two arguments:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• get: downloads a text or binary file or creates a query from the contents of a text file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• post: sends information to the server page or CGI program for processing. Requires the params argument.</td>
</tr>
<tr>
<td>url</td>
<td>Required</td>
<td>The absolute URL of the host name or IP address of the server on which the file resides. The URL must include the protocol (http or https) and host name.</td>
</tr>
<tr>
<td>username</td>
<td>Optional</td>
<td>When required by a server, a username.</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td>When required by a server, a password.</td>
</tr>
</tbody>
</table>
| resolveurl | Optional  | For GET and POST methods.  
  • Yes or No. Default is No.  
For GET and POST operations, if Yes, the page reference that is returned into the Filecontent property has its internal URLs fully resolved, including port number, so that links remain intact. The following HTML tags, which can contain links, are resolved:  
- img src  
- a href  
- form action  
- applet code  
- script src  
- embed src  
- embed pluginspace  
- body background  
- frame src  
- bgsound src  
- object data  
- object classid  
- object codebase  
- object usemap |
<table>
<thead>
<tr>
<th><strong>params</strong></th>
<th><strong>Optional</strong></th>
<th><strong>HTTP parameters passed as an array of objects. Supports the following parameter types:</strong></th>
</tr>
</thead>
</table>
|             |              | • name  
|             |              | • type  
|             |              | • value  
|             |              | **CF.http** params are passed as an array of objects. The **params** argument is required for POST operations. |
| **path**    | **Optional** | **The path to the directory in which to store files. When using the path argument, the file argument is required.** |
| **file**    | **Optional** | **Name of the file that is accessed. For GET operations, defaults to the name specified in the url argument. Enter path information in the path argument. This argument is required if you are using the path argument.** |

**Usage**

You can write the `CF.http` function using named arguments or positional arguments. You can invoke all supported arguments using the named argument style, as follows:

```coldfusion
CF.http({method:"method", url:"URL", username:"username", password:"password", resolveurl:"yes or no", params:arrayvar, path:"path", file:"filename"});
```

⚠️ The named argument style uses curly braces () to surround the function arguments.

Positional arguments let you use a shorthand coding style. However, not all arguments are supported for the positional argument style. Use the following syntax to code the `CF.http` function using positional arguments:

```coldfusion
CF.http(url);  
CF.http(method, url);  
CF.http(method, url, username, password);  
CF.http(method, url, params, username, password);  
```
Do not use curly braces `{}` with positional arguments.

The following parameters can only be passed as an array of objects in the `params` argument in the `CF.http` function:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The variable name for data that is passed</td>
</tr>
</tbody>
</table>
| type      | The transaction type:  
  - URL  
  - FormField  
  - Cookie  
  - CGI  
  - File |
| value     | Value of URL, FormField, Cookie, File, or CGI variables that are passed |

The `CF.http` function returns data as a set of object properties, as described in the following table:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>A Boolean value that indicates whether the specified URL location contains text data.</td>
</tr>
</tbody>
</table>
| Charset  | The charset used by the document specified in the URL.  
  HTTP servers normally provide this information, or the charset is specified in the charset parameter of the Content-Type header field of the HTTP protocol. For example, the following HTTP header announces that the character encoding is EUC-JP:  
  `Content-Type: text/html; charset=EUC-JP` |
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td>Raw response header. For example:</td>
</tr>
<tr>
<td></td>
<td>HTTP/1.1 200 OK</td>
</tr>
<tr>
<td></td>
<td>Date: Mon, 04 Mar 2002 17:27:44 GMT</td>
</tr>
<tr>
<td></td>
<td>Server: Apache/1.3.22 (Unix) mod_perl/1.26</td>
</tr>
<tr>
<td></td>
<td>Set-Cookie: MM_cookie=207.22.48.162.4731015262864476; path=/; expires=Wed, 03-Mar-04 17:27:44 GMT; domain=adobe.com</td>
</tr>
<tr>
<td></td>
<td>Connection: close</td>
</tr>
<tr>
<td></td>
<td>Content-Type: text/html</td>
</tr>
<tr>
<td><strong>Filecontent</strong></td>
<td>File contents, for text and MIME files.</td>
</tr>
<tr>
<td><strong>Mimetype</strong></td>
<td>MIME type. Examples of MIME types include text/html, image/png, image/gif, video/mpeg, text/css, and audio/basic.</td>
</tr>
<tr>
<td><strong>responseHeader</strong></td>
<td>Response header. If there is only one header key, its value can be accessed as simple type. If there are multiple header keys, the values are put in an array in a responseHeader structure.</td>
</tr>
<tr>
<td><strong>Statuscode</strong></td>
<td>HTTP error code and associated error string. Common HTTP status codes returned in the response header include:</td>
</tr>
<tr>
<td></td>
<td>400: Bad Request</td>
</tr>
<tr>
<td></td>
<td>401: Unauthorized</td>
</tr>
<tr>
<td></td>
<td>403: Forbidden</td>
</tr>
<tr>
<td></td>
<td>404: Not Found</td>
</tr>
<tr>
<td></td>
<td>405: Method Not Allowed</td>
</tr>
</tbody>
</table>

You access these attributes using the `get` function:
function basicGet()
{
  url = "http://localhost:8100/";

  // Invoke with just the url. This is an HTTP GET.
  result = CF.http(url);
  return result.get("Filecontent");
}

For more information on using server-side ActionScript, see Using Server-Side ActionScript in the Developing ColdFusion Applications.

**Example**

The following examples show a number of the ways to use the CF.http function:
function postWithNamedArgs()
{
    // Set up the array of Post parameters.
    params = new Array();
    params[1] = {name: "arg1", type: "FormField", value: "value1"};

    url = "http://localhost:8100/";

    path = application.getContext("/").getRealPath("/");
    file = "foo.txt";


    if (result)
        return result.get("Statuscode");
    return null;
}

// Example of a basic HTTP GET operation
// Shows that HTTP GET is the default
function basicGet()
{
    url = "http://localhost:8100/";

    // Invoke with just the url. This is an HTTP GET.
    result = CF.http(url);
    return result.get("Filecontent");
}

// Example showing simple array created to pass params arguments
function postWithParams()
{
    // Set up the array of Post parameters. These are just like cfhttpparam tags.
    params = new Array();
    params[1] = {name: "arg2", type: "URL", value: "value2"};

    url = "http://localhost:8100/";

    // Invoke with the method, url, and params
    result = CF.http("post", url, params);
    return result.get("Filecontent");
}

// Example with username and params arguments
function postWithParamsAndUser()
{
    // Set up the array of Post parameters. These are just like cfhttpparam tags.
    params = new Array();
    params[1] = {name: "arg2", type: "URL", value: "value2"};

    url = "http://localhost:8100/";

    // Invoke with the method, url, params, username, and password
    result = CF.http("post", url, params, "karl", "salsa");
    return result.get("Filecontent");
}
**CF.query**

*Description*

Performs queries against ColdFusion data sources.

*Return value*

Returns a RecordSet object.

*Syntax*

```coldfusion
CF.query({
      datasource:"data source name",
      sql:"SQL stmts",
      username:"username",
      password:"password",
      maxrows:number,
      timeout:milliseconds
})
```

*Arguments*

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Req/Opt</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>Required</td>
<td>Name of the data source from which the query retrieves data.</td>
</tr>
<tr>
<td>sql</td>
<td>Required</td>
<td>SQL statement.</td>
</tr>
<tr>
<td>username</td>
<td>Optional</td>
<td>Username. Overrides the username specified in the data source setup.</td>
</tr>
<tr>
<td>password</td>
<td>Optional</td>
<td>Password. Overrides the password specified in the data source setup.</td>
</tr>
<tr>
<td>maxrows</td>
<td>Optional</td>
<td>Maximum number of rows to return in the record set.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional</td>
<td>Maximum number of seconds for the query to execute before returning an error indicating that the query has timed out. Can only be used in named arguments.</td>
</tr>
</tbody>
</table>
**Usage**

You can code the `CF.query` function using named or positional arguments. You can invoke all supported arguments using the named argument style, as follows:

```cfm
CF.query({datasource:"datasource", sql:"sql stmt", 
    username:"username", password:"password", maxrows:"maxrows", 
    timeout:"timeout"});
```

⚠️ The named argument style uses curly braces {} to surround the function arguments.

Positional argument style, which is a shorthand coding style, does not support all arguments. Use the following syntax to code the `CF.query` function using positional arguments:

```cfm
CF.query(datasource, sql);
CF.query(datasource, sql, maxrows);
CF.query(datasource, sql, username, password);
CF.query(datasource, sql, username, password, maxrows);
```

⚠️ Do not use curly braces {} with positional arguments.

You can manipulate the record set returned by the `CF.query` function using methods in the RecordSet ActionScript class. The following are some of the methods available in the RecordSet class:

- `RecordSet.getColumnnames`
- `RecordSet.getLength`
- `RecordSet.getItemAt`
- `RecordSet.getItemId`
- `RecordSet.sortItemsBy`
- `RecordSet.getNumberAvailable`
- `RecordSet.filter`
- `RecordSet.sort`

For more information on using server-side ActionScript, see Using Server-Side ActionScript in the Developing ColdFusion Applications. For more detailed information about the RecordSet ActionScript class, see Using Flash Remoting.

**Example**

---

© 2014 Adobe Systems Incorporated. All rights reserved.
// Define a function to do a basic query
// Note use of positional arguments
function basicQuery()
{
    result = CF.query("myquery", "cust_data", "SELECT * from tblParks");
    return result;
}

// Example function declaration using named arguments
function basicQuery()
{
    result = CF.query({datasource:"cust_data", sql:"SELECT * from tblParks"});
    return result;
}

// Example of the CF.query function using maxrows argument
function basicQueryWithMaxRows()
{
    result = CF.query("cust_data", "SELECT * from tblParks", 25);
    return result;
}

// Example of the CF.query function with username and password
function basicQueryWithUser()
{
    result = CF.query("cust_data", "SELECT * from tblParks", "wsburround", "migraine1");
    return result;
}
ColdFusion Mobile Functions

- Accelerometer Functions
- Camera Functions
- Connection Functions
- Contact Functions
- Event Functions
- File System Functions
- Geolocation Functions
- Media and Capture Functions
- Notification Functions
- Splash Screen Functions
- Storage Functions
Accelerometer Functions

The Accelerometer API allows you to capture the device motion in the X, Y, and Z direction. The ColdFusion Accelerometer API dispatches certain events based on the activity detected by the device’s motion sensor (accelerometer). The data returned by the API represents the device’s location or movement along a 3-dimensional axis. When the device moves, the sensor detects this movement and returns acceleration data.

- `cfclient.accelerometer.clearWatch`
- `cfclient.accelerometer.getOptions`
- `cfclient.accelerometer.setOptions`
- `cfclient.accelerometer.watch`
**cfclient.accelerometer.clearWatch**

*Stop watching position changes*

You can stop watching the device position using the `clearWatch()` function.

**Method summary**

- **returns** – None
- **params**
  - `watchID` – String

For instance,

```
<cfset cfclient.accelerometer.clearWatch(watchId)>
```

**Example**

See [Using the Accelerometer APIs](#)
**cfclient.accelerometer.getOptions**

*Getting Accelerometer options*

You can get the Accelerometer options using the getOptions() function

**Method summary**

<table>
<thead>
<tr>
<th>Method summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>getOptions()</strong></td>
<td>returns Options object, params None</td>
</tr>
</tbody>
</table>

For instance,

```coldfusion
<cfset opt = cfclient.accelerometer.getOptions()>
```

**The Options object**

The Options object contains the following parameter:

- **frequency** – The time delay for invoking the callback function to obtain the Accelerometer data (in milliseconds)

**Example**

See [Using the Accelerometer APIs](#)
**cfclient.accelerometer.setOptions**

*Setting Accelerometer options*

You can also set the Accelerometer options using the `setOptions()` function.

**Method summary**

<table>
<thead>
<tr>
<th>setOptions(options)</th>
</tr>
</thead>
</table>

- **returns** – None
- **params**
  - Options object

For instance,

```html
<cfset opt.frequency=20000>
<cfset cfclient.accelerometer.setOptions(opt)>
```

**The Options object**

The Options object contains the following parameter:

- **frequency** – The time delay for invoking the callback function to obtain the Accelerometer data (in milliseconds)

**Example**

See [Using the Accelerometer APIs](#)
cfclient.accelerometer.watch

Watching for change in position
You can get the acceleration data (X, Y, and Z) at regular intervals by watching for the change in device position.

Method summary

<table>
<thead>
<tr>
<th>returns</th>
<th>watchID - String</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td></td>
</tr>
<tr>
<td>callback</td>
<td>Function that will be called whenever there is a change in location</td>
</tr>
<tr>
<td>Options object</td>
<td>(Optional)</td>
</tr>
</tbody>
</table>

For instance,

```cfc
<cfset watchId=cfclient.accelerometer.watch(callbackfunc)>
```

The returned watch ID references the accelerometer's watch interval, and can be used with accelerometer.clearWatch to stop watching the accelerometer.

```cfc
<cffunction access="public" name="callbackfunc" returntype="void">
<cfargument name="acceleration">
<!--- Call back function code --->
<cfset writeOutput("acceleration change X-" & obj.x & " Y-" & obj.y & " Z-" & obj.z & " Timestamp -" & obj.timestamp )><br/>
</cffunction>
```

The callback function passed to the watch function will be invoked based on the frequency provided in the options object. The callback function will receive an argument 'acceleration' with the following properties:

- x - Amount of acceleration on the x-axis
- y - Amount of acceleration on the x-axis
- z - Amount of acceleration on the z-axis
- timestamp - The creation timestamp in milliseconds

The 'acceleration' object has the accelerometer data captured at a specific time.

The Options object

The Options object contains the following parameter:

- frequency – The time delay for invoking the callback function to obtain the Accelerometer data (in milliseconds)

Example
See Using the Accelerometer APIs
Camera Functions

ColdFusion allows you to access the camera of the mobile device through simple CFML code.

The following sections describe how you can invoke the camera using `<cfclient>`.

- `<cfclient.camera.cleanup>`
- `<cfclient.camera.getOptions>`
- `<cfclient.camera.getPicture>`
- `<cfclient.camera.getPictureFromAlbum>`
- `<cfclient.camera.getPictureFromPhotoLibrary>`
- `<cfclient.camera.setOptions>`
- `<options Object>`

ℹ️ The image quality of pictures taken using the camera on newer devices is quite good, and images from the Photo Album will not be downscaled to a lower quality, even if a quality parameter is specified. Encoding such images using Base64 can cause memory issues on many newer devices. Therefore, using FILE_URI for images captured is highly recommended.
**cfclient.camera.cleanup**

*Removing intermediate photos*

You can remove the intermediate photos taken by the camera from the temporary storage.

```coldfusion
<cfset cfclient.camera.cleanup()>
```

Note: This function is supported only on iOS devices.

Method summary

**cleanup**

- **returns** – None
- **params**
  - *None*

**Example**

See [Using the Camera APIs](#)
**cfclient.camera.getOptions**

*Getting the camera options*

You can get the camera options to customize the camera settings using the `getOptions()` function as shown below:

```coldfusion
<cfset option = cfclient.camera.getOptions()>
```

**Method summary**

`getOptions`

- **returns** – Returns the [Options](#) object
- **params**
  - None

**Example**

See [Using the Camera APIs](#)
**cfclient.camera.getPicture**

The image quality of pictures taken using the camera on newer devices is quite good, and images from the Photo Album will not be downscaled to a lower quality, even if a quality parameter is specified. Encoding such images using Base64 can cause memory issues on many newer devices. Therefore, using FILE_URI for images captured is highly recommended.

Getting the image from the camera

By invoking the default camera application on the device, this function takes a photo using the camera as the source.

The following example shows how you can get an image from the camera using the getPicture function:

```cfclient
<cfset picture = cfclient.camera.getPicture(options, base64)>
```

The following example depicts the usage:

```cfclient
<cfclientsettings enableDeviceAPI=true>
<cfclient>
<cffunction access="public" name="getpiccam" returntype="void">
<cfset opt = cfclient.camera.getOptions()>
<cfset response = cfclient.camera.getPicture(opt,true)>
<cfset response = 'data:image/jpeg;base64,' + response>
<cfset src_to_img("myimg", response)>
</cffunction>
</cfclient>

<button style="width:100;height:50" onclick="invokeCFClientFunction('getpiccam',null)">Get a picture</button>

<img style = "width:200; height:100;" id = "myimg"></img>

<script type="text/javascript">
function src_to_img(img_id, data_to_write) {
    document.getElementById(img_id).src=data_to_write;
}
</script>
```

In the above example, on the click of a button, the default camera application is opened and after capturing an image, the image is rendered in the `<img>` tag.

Note: If you pass true as a param indicating that the returned object should be a Base-64 encoded string, its content
has to be prefixed with “data:image/jpeg;base64,” before rendering it.

Method summary

`getPicture(Options, base64)`

- **returns** – Returns the image file URI or a Base-64 encoded string
- **params**
  - `Options` – Object
  - `base64` – A Boolean flag to indicate that the returned object should be a base-64 encoded string.

Format for the Options object

The following table has the list of the properties for the Options object (ColdFusion sets defaults for these properties):

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>allowEdit</code></td>
<td>Boolean</td>
</tr>
<tr>
<td><code>encodingType</code></td>
<td>JPEG or PNG</td>
</tr>
<tr>
<td><code>popoverOptions</code></td>
<td>Object</td>
</tr>
<tr>
<td><code>savePhotoToAlbum</code></td>
<td>boolean</td>
</tr>
<tr>
<td><code>Quality</code></td>
<td>Number (0-100)</td>
</tr>
</tbody>
</table>

For all supported properties, check this document.

*Example*

See Using the Camera APIs
cfclient.camera.getPictureFromAlbum

Getting the image from the album

Using the getPictureFromAlbum function, you can get the image stored in the album.

Method summary

getPictureFromAlbum(Options, [base64])

- **returns** – Returns the file URI or a base-64 encoded string
- **params**
  - **Options** – Object
  - **base64** – A Boolean flag to indicate that the returned object should be a base-64 encoded string. The default value is false. When it is false, it will return the file URI.

**Note:** If you pass true as a parameter indicating that the returned object should be a Base-64 encoded string, its content has to be prefixed with “data:image/jpeg;base64,” before rendering it.

The following example shows how you can get an image from the local file system using the getPictureFromAlbum function:

```cfclientsettings enableDeviceAPI=true>
<cfclient>
<cffunction access="public" name="getPicAlbm" returnType="void">
<cfset opt = cfclient.camera.getOptions()>
<cfset response = cfclient.camera.getPictureFromAlbum(opt,true)>

<cfset response = 'data:image/jpeg;base64,' + response>
<cfset src_to_img("myimg", response)>
</cffunction>
</cfclient>
</cfclientsettings>
<button style="width:100;height:50"
onclick="invokeCFClientFunction('getPicAlbm',null)">Get a picture</button>

<image style = "width:200; height:100;" id = "myimg"></image>

<script type="text/javascript">
function src_to_img(img_id,data_to_write)
{
    document.getElementById(img_id).src=data_to_write;
}
</script>

**Example**

See Using the Camera APIs
cfclient.camera.getPictureFromPhotoLibrary

Getting the image from the photo library

Using the getPictureFromPhotoLibrary function, you can get the image stored in the photo library.

Method summary

getPictureFromPhotoLibrary(Options, base64)

- **returns** – Returns the file URI or a base-64 encoded string
- **params**
  - **Options** – Object
  - **base64** – A Boolean flag to indicate that the returned object should be a base-64 encoded string.

**Note:** If you pass true as a param indicating that the returned object should be a Base-64 encoded string, its content has to be prefixed with “data:image/jpeg;base64,” before rendering it.

The following example shows how you can get an image from the local file system using the getPictureFromPhotoLibrary function:

```plaintext
<cfclientsettings enableDeviceAPI=true>
<cfclient>
<cffunction access="public" name="getPicCam" returntype="void" >
<cfset opt = cfclient.camera.getOptions()>
<cfset response = cfclient.camera.getPictureFromPhotoLibrary (opt,true)>
<cfset response = 'data:image/jpeg;base64,' + response>
<cfset src_to_img("myimg", response)>
</cffunction>
</cfclient>
</cfclientsettings>

<button style="width:100;height:50" onclick="invokeCFClientFunction('getPicCam',null)">Get a picture</button>

<img style = "width:200; height:100;" id = "myimg"></img>

<script type="text/javascript">
function src_to_img(img_id,data_to_write)
{
    document.getElementById(img_id).src=data_to_write;
}
</script>

Example

See Using the Camera APIs
cfclient.camera.setOptions

Setting the camera options

You can set the options to customize the camera settings using the setOptions() function as shown below:

```cfml
<cfset options = {"quality" = "40", "encodingType" = "png","saveToPhotoAlbum":true}>
<cfset cfclient.camera.setOptions(options)>
```

Method summary

setOptions(options)

- returns – None
- params
  - Options object

Example

See Using the Camera APIs
options Object

Format for the Options object

The following table has the list of properties for which ColdFusion sets defaults:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>encodingType</td>
<td>JPEG or PNG</td>
</tr>
<tr>
<td>quality</td>
<td>Number (0-100)</td>
</tr>
<tr>
<td>height</td>
<td>Number</td>
</tr>
<tr>
<td>width</td>
<td>Number</td>
</tr>
</tbody>
</table>

For a complete list of supported properties, see this document.
Connection Functions

The connection API allows you to detect the type of phone connection. Using these APIs, you can find out the connection type (2G, 3G, or 4G) and register for events when the phone goes online or offline.

- `cfclient.connection.getType`
- `cfclient.connection.onOffline`
- `cfclient.connection.onOnline`
cfclient.connection.getType

**Getting the type of connection**

You can get the connection type of the device using the `getType()` function.

**Method summary**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getType()</code></td>
<td>Method returns - Number</td>
</tr>
</tbody>
</table>

- **returns** - Number

- **params**
  - None

For instance,

```
<cfset conType = cfclient.connection.getType()>
```

The connection types returned by these functions will be one of the following constants:

- `cfclient.connection.UNKNOWN`
- `cfclient.connection.ETHERNET`
- `cfclient.connection.WIFI`
- `cfclient.connection.CELL_2G`
- `cfclient.connection.CELL_3G`
- `cfclient.connection.CELL_4G`
- `cfclient.connection.NONE`
**cfclient.connection.onOffline**

*Getting notification when the device goes offline*

You can register a callback function to be invoked when the device goes offline using the `onOffline()` function.

**Method summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>onOffline</code></td>
<td><code>onOffline(callbackFunction)</code></td>
</tr>
</tbody>
</table>

- **returns** – None
- **params**
  - Name of the callback function that gets invoked when the device goes offline.

For instance,

```cfc
<cfset cfclient.connection.onOffline(onofflinecallback)>
<cffunction name="'onofflinecallback'" >
  <!--- Code to be executed when device goes offline --->
</cffunction>
```
**cfclient.connection.onOnline**

*Getting notification when the device goes online*

You can register a callback function to be invoked when the device goes online using the onOnline() function.

**Method summary**

<table>
<thead>
<tr>
<th>onOnline(callBackFunction)</th>
</tr>
</thead>
</table>

- **returns** – None
- **params**
  - Name of the callback function that gets invoked when the device goes online.

For instance,

```cfml
<cfsset cfclient.connection.onOnline(onlinecallback)>
<cffunction name="ononlinecallback" >
  <!--- Code to be executed when device goes online --->
</cffunction>
```
Contact Functions

ColdFusion allows you to access the contacts of the mobile device through simple CFML code.

The following sections describe how you can work with phone contacts using `cfclient`.

- Contact Creation Functions
- Contacts Searching Functions
Contact Creation Functions

Creating a new contact

You can use create, createName, createAddress, and createOrganization to add a new contact to the phonebook.

Method summary

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Method summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Contact object with the given data.</td>
<td><code>create ()</code></td>
</tr>
<tr>
<td></td>
<td>• Returns – Contact Object</td>
</tr>
<tr>
<td></td>
<td>• Params</td>
</tr>
<tr>
<td></td>
<td>• <code>name</code> – String</td>
</tr>
<tr>
<td></td>
<td>• <code>phoneNumber</code> – String</td>
</tr>
<tr>
<td></td>
<td>• <code>email</code> – String</td>
</tr>
<tr>
<td></td>
<td>• <code>save</code> – boolean flag to indicate whether to save the contact or not.</td>
</tr>
<tr>
<td>Create ContactName object with the given data.</td>
<td><code>createName(String givenName, String familyname, String middleName, String formatted, String prefix, String suffix)</code></td>
</tr>
<tr>
<td></td>
<td>• Returns - ContactName Object</td>
</tr>
<tr>
<td></td>
<td>• Params</td>
</tr>
<tr>
<td></td>
<td>• <code>ContactName</code> - represents Name for given contact</td>
</tr>
<tr>
<td></td>
<td>• Fields available for contact</td>
</tr>
<tr>
<td></td>
<td>• <code>formatted</code></td>
</tr>
<tr>
<td></td>
<td>• <code>familyname</code></td>
</tr>
<tr>
<td></td>
<td>• <code>givenname</code></td>
</tr>
<tr>
<td></td>
<td>• <code>middlename</code></td>
</tr>
<tr>
<td></td>
<td>• <code>honorificprefix</code></td>
</tr>
<tr>
<td></td>
<td>• <code>honorificsuffix</code></td>
</tr>
</tbody>
</table>
Create ContactAddress object with the given data.

```java
createAddress(String streetAddress, String locality, String region, String country, String postalCode, String formatted, String type, Boolean pref)
```

- Returns - ContactAddress object
- Params
  - `streetAddress` - String
  - `locality` – String
  - `region` – String
  - `country` – String
  - `postalCode` – String
  - `formatted` – String
  - `type` – String
  - `pref` - boolean flag to specify whether given contact field should be considered as preferred number

Create ContactOrganization object with the given data.

```java
createOrganization(String name, String department, String title, String type, Boolean pref)
```

- Returns - ContactOrganization object
- Params
  - `name` - String
  - `department` – String
  - `title` – String
  - `type` – String
  - `pref` - boolean flag to specify whether given contact field should be considered as preferred number

Some examples

To create a Contact object:

```xml
<cfset contactobj =
cfclient.contacts.create('user','123456789','user@adobe.com',true)>
```

To create a name:

```xml
<cfset cntname=cfclient.contacts.createName("Rob", "Cressey", "T", "Rob T Cressey", "Mr" , "Esq.")>
```

To create an address:
<cfset cntadr=cfclient.contacts.createAddress("First street", "North", "Indian", "India","12345", "Good address form", "home", "true")>

To create an organization:

Contacts Searching Functions

Searching a contact

You can use the find function to search through the contacts list with a criterion.

Method summary

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Method summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find contacts based on the filter options</td>
<td>find(String filter, Array fields)</td>
</tr>
<tr>
<td></td>
<td>• Returns – An array of Contact objects. If the fields are specified, the returned Contact object will contain values for only those fields.</td>
</tr>
<tr>
<td></td>
<td>• Params</td>
</tr>
<tr>
<td></td>
<td>• filter – A string to specify the filter for search.</td>
</tr>
</tbody>
</table>
|                                        | • fields - Array of fields to include in the returned Contact Object. Note that the search will be performed only for the specified fields. Valid values include name, displayName, phoneNumbers, and *.

An example

To find a contact “Rob” matching the name and displayName:

```cftmpl
<cfset cntlst=cfclient.contacts.find("Rob", ["name","displayName"])>
<cfscript>
  for (i = 1; i <= #arrayLen(cntlst)#; i = i + 1) {
    /*Populating the content of the <div> with the ID divId*/
    document.getElementById('divId').innerHTML=cntlst[i].displayName;
  }
</cfscript>
```

Getting all contact information

To quickly get a list of all contacts, use the getAllContacts function as shown in the following example:

Method summary

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Method summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove the contact matching filter</td>
<td></td>
</tr>
</tbody>
</table>

© 2014 Adobe Systems Incorporated. All rights reserved.
Get all the stored contacts

### `getAllContacts`

- Returns – an array of Contact objects. If the fields are specified, the returned contact object contains values only for those fields.
- **Params**
  - **fields** – an array of fields to include in the returned Contact Object. The valid values include name, displayName, phoneNumbers, and *(indicates all fields).

### An example

To enumerate all the contacts:

```coldfusion
<cfset cntlst=cfclient.contacts.getAllContacts(["name")])>
<cfscript>
  for (i = 1; i <= #ArrayLen(cntlst)#; i = i + 1) {
    /*Populating the content of the <div> with the ID divId*/
    document.getElementById('divId').innerHTML=
    cntlst[i].name.givenName;
  }
</cfscript>
```

### The Contact object

The following list describes the properties available for the Contact object:

- id
- displayName
- name (ContactName)
- nickName
- phoneNumbers (ContactField [ ])
- emails (ContactField [ ])
- ims (ContactField [ ])
- organizations
- birthday
- note
- photos (ContactField [ ])
- categories (ContactField [ ])
- urls (ContactField [ ])

The following APIs are supported only on Android devices. Some of these APIs may not work on iOS devices.

The following list describes the functions available for the Contacts object:

- `getMobileNumber()`
The Contact Object supports a few additional methods. For more information about the supported methods, read this document.
Event Functions

You can let your applications listen to the device events by adding specific event listeners. The following sections show you how to handle device events.

- `cfclient.events.onBackButton`
- `cfclient.events.onBatteryCritical`
- `cfclient.events.onBatteryLow`
- `cfclient.events.onBatteryStatusChange`
- `cfclient.events.onMenuButton`
- `cfclient.events.onPause`
- `cfclient.events.onResume`
- `cfclient.events.onSearchButton`
**cfclient.events.onBackButton**

*Handling the Back button event*

To handle the device back button, use:

```cfc
<cfset cfclient.events.onBackButton(callbackfunc)>
```
cfclient.events.onBatteryCritical

Handling the critical battery event

To handle the battery critical event, use:

```plaintext
<cfset cfclient.events.onBatteryCritical(callbackfunc)>
```
cfclient.events.onBatteryLow

Handling the low battery event

To handle the low battery event, use:

```cfc不多 import cfclient.events
<cfset cfclient.events.onBatteryLow(callbackfunc)>```
Handling the battery status change event

To handle the battery status change event, use:

```
<cfset cfclient.events.onBatteryStatusChange(callbackfunc)>
```
**cfclient.events.onMenuButton**

*Handling the Menu button event*

To handle the Menu button event, use:

```
<cfset cfclient.events.onMenuButton(callbackfunc)>
```
cfclient.events.onPause

Handling the pause event

To handle the device pause event, use:

```cfc
<cfset cfclient.events.onPause(callbackfunc)>
```
cfclient.events.onResume

Handling the resume event

To handle the device resume event, use:

```coldfusion
<cfset cfclient.events.onResume(callbackfunc)>
```
cfclient.events.onSearchButton

Handling the Search button event

To handle the Search button event, use:

```cfset cfclient.events.onSearchButton(callbackfunc)>
```

For all battery related events, the callback function is invoked with a battery status object. The following properties are associated with this object:

- **level** - The percentage of battery (0-100). (Number)
- **isPlugged** - A boolean that represents whether or not the device is plugged in.
File System Functions

ColdFusion allows you to access the file system of the mobile device through simple CFML code.

The following sections describe how you can manage the native file system using `<cfclient>`.

- `cfclient.file.append`
- `cfclient.file.copy`
- `cfclient.file.copyDirectory`
- `cfclient.file.createDirectory`
- `cfclient.file.directoryExists`
- `cfclient.file.download`
- `cfclient.file.exists`
- `cfclient.file.get`
- `cfclient.file.getDirectory`
- `cfclient.file.getWorkingDirectory`
- `cfclient.file.listdir`
- `cfclient.file.move`
- `cfclient.file.moveDirectory`
- `cfclient.file.read`
- `cfclient.file.readAsBase64`
- `cfclient.file.remove`
- `cfclient.file.removeDirectory`
- `cfclient.file.renameDirectory`
- `cfclient.file.setFileSystem`
- `cfclient.file.setWorkingDirectory`
- `cfclient.file.upload`
- `cfclient.file.write`
- `DirectoryEntry Object`
- `FileEntry Object`
- `rename Function`

**A generic note**

All the file system functions support file URL (file path starting with `file://`) apart from supporting absolute and relative paths.
**cfclient.file.append**

*Appending to a file*

You can append text to a file using the append function.

**Method summary**

`append (name,data) - append data to a given file`

- **returns** - None
- **params**
  - name – Absolute or relative path of a file name (including file name)
  - data - data to be written to file

For instance,

```coldfusion
<cfset cfclient.file.write('first.txt','Hello there!!!')>
<cfset cfclient.file.append('first.txt','Adding more text.' )>
```

**Examples**

See [Using the File APIs](#)
cfclient.file.copy

Copying a file

You can copy a file from one location to another location on the device by using the copy function as shown in the following example:

```
<cfset cfclient.file.copy(source,destination)>
```

Method summary

copy(source, destination) – copies a file from source to destination

- **returns** - None
- **params**
  - source – Absolute or relative path of the source filename (including the file name)
  - destination – Absolute or relative path of the destination filename (including the file name)

Examples

See Using the File APIs
**cfclient.file.copyDirectory**

*Copying a directory*

You can copy a directory by using the `copyDirectory` function as shown in the following example:

```cfc
<cfset cfclient.file.copyDirectory(source,destination)>
```

### Method summary

`copyDirectory (source, destination)` - copy directory from source to destination

- **Returns** – None
- **Params**
  - `source` – The source directory name with absolute or relative path
  - `destination` – Absolute or relative path of the destination location (excluding directory name)

For instance,

```cfc
<cfset cfclient.file.copyDirectory('dir2','dir1')>
```

### Examples

See [Using the File APIs](#)
**cfclient.file.createDirectory**

*Creating a directory*

You can create a directory on the device as shown in the following code example:

```cf
<cfset newdir = "myNewDir" />  
<cfset dirobj=cfclient.file.createDirectory(newdir)>
```

You can specify both absolute and relative path for the new directory. By default, the directory will be created on the device’s memory card.

**Method summary**

`createDirectory(dirPath, [throwError])`

- **returns** DirectoryEntry object
- **params**
  - `dirPath` – Relative/Absolute path of the directory to be created (including the directory name).
  - `throwError` - A boolean value indicating whether an error should be shown when the directory exists.

*Reading the properties of file and directory objects*

The following sample code shows how you can read directory object properties:

```cf
<cfclient>
    <cfset dirobj=cfclient.file.createDirectory('myNewDir')>
    <cfset evalStr = "document.getElementById('divId').innerHTML='" &
    dirobj.fullPath & '"">
    <cfset eval(#evalStr#)>
</cfclient>
```

**Examples**

See [Using the File APIs](#)
cfclient.file.directoryExists

Checking if the directory exists

You can check if the directory exists by using the directoryExists function as shown in the following example:

```
<cfset existresponse=cfclient.file.directoryExists("myNewDir")>
<cfif existresponse >
   <!---Directory Exists --->
</cfif>
```

Method summary

directoryExists (path)

- **Params**
  - **path** An absolute or relative path of the directory (including directory name).

- **Returns** – a Boolean value indicating the existence of the specified directory
cfclient.file.download

Download a file

You can download a file from a remote server.

Method summary

cfclient.file.download(url,localFilePath,successCallback, errorCallback,[options])

- **returns** - FileEntry object
- **params**
  - url - The remote server URL (CFM page)
  - localFilePath – Absolute or relative path of a file name (including file name)
  - successCallback - The function to be invoked upon success/completion
  - errorCallback - The function to be invoked if there is a file transfer error
  - options object (see below for the supported options)

Options object

Optional parameters, currently only supports headers (such as Authorization (Basic Authentication), etc).

Example

See [Getting Started Examples](#)
**cfclient.file.exists**

*Checking if the file exists*

You can determine if the specified file exists by using the exists function as shown in the following example:

```coldfusion
<cfset existresponse = cfclient.file.exists("myNewFile.txt")>
```

**Method summary**

`exists(filePath)` - check if file exists at specified location

- **returns** - true/false
- **params** - filePath  Absolute or relative path of the file to be checked (including file name).
**cfclient.file.get**

*Looking up the file*

You can look up for the file using the get function as shown in the following example:

```coldfusion
cfset fileobj=cfclient.file.get("myNewFile.txt")
```

**Method summary**

*get(filePath) – Lookup for the file in the given path*

- **returns** – A FileEntry object
- **params**
  - `filePath` – An absolute or relative path of the file (including the file name).
**cfclient.file.getDirectory**

*Looking up the directory*

You can look up for the directory using the getDirectory function as shown in the following example:

```
<cfset dirobj=cfclient.file.getDirectory("myNewDir")>
```

**Method summary**

`getDirectory(dirPath)` — Lookup for the directory in the given path

- **returns** — A DirectoryEntry object
- **params**
  - `dirPath` — An absolute or relative path of the directory (including directory name).
**cfclient.file.getWorkingDirectory**

*Getting the current directory*

You can get the current working directory by using the `getWorkingDirectory()` function.

**Method summary**

`getWorkingDirectory()` – *Lookup for the current working directory*

- **returns** – A String containing the path to the current working directory
- **params**
  - None.

For instance,

```
<cset pwd=cfclient.file.getWorkingDirectory()>
```
**cfclient.file.listDirectory**

*Listing a directory*

You can list the content of a directory using the listDirectory function.

```coldfusion
<cfset dircontent= cfclient.file.listDirectory("myNewDir")>
```

**Method summary**

`listDirectory(dirPath)` - lists contents of directory specified by `dirPath`

- **returns** – An array of FileEntry and DirectoryEntry objects
- **params**
  - `dirPath` – Absolute or relative path of the directory to be listed (including the directory name).
  - `sortType` - A string representing the type of directory sorting. For instance ("name ASC/DESC", "datemodified ASC/DESC")
**cfclient.file.move**

**Moving a file**

You can move a file from one location to another location on the device by using the move function as shown in the following example:

```xml
<cfset fileobj=cfclient.file.move(source,destination)>
```

**Method summary**

*move(source, destination) – moves a file from source to destination*

- **returns** - FileEntry object
- **params**
  - source – Absolute or relative path of the source filename (including the file name)
  - destination – Absolute or relative path of the destination filename (including the file name)


**cfclient.file.moveDirectory**

*Moving a directory*

You can move a directory by using the moveDirectory function as shown in the following example:

```cfml
<cfset dirobj = cfclient.file.moveDirectory(source, destination)>
```

**Method summary**

`moveDirectory(source, destination)` - *move a directory from source to destination*

- **returns** - `DirectoryEntry` object
- **params**
  - source – Source directory name (absolute or relative path)
  - destination – Absolute or relative path of the destination location (excluding the directory name)

For instance,

```cfml
<cfset cfclient.file.moveDirectory('dir2','dir1')>
```
cfclient.file.read

*Reading a file*

You can read a text file with the read function. The following example shows a simple usage:

```cfml
<cfset filecontent=cfclient.file.read("myNewFile.txt")>
```

**Method summary**

`read(filename)` - *Reads the content of a given file*

- **returns** – Content of file in the form of a string
- **params**
  - `filename` - Absolute or relative path of the file (including file name)
**cfclient.file.readAsBase64**

*Reading a file as a Base64 string*

You can read a text file with the read function. The following example shows a simple usage:

```cfml
<cfset opt = cfclient.camera.getOptions()>
<cfset res = cfclient.camera.getPicture(opt,false)>
<cfset imgStr=cfclient.file.readAsBase64(res)>
```

**Method summary**

`readAsBase64(filename)` - *Reads the content of a given file*

- **returns** – Content of file in the form of a base64 encoded string
- **params**
  - `filename` - Absolute or relative path of the file (including file name)
cfclient.file.remove

Deleting a file

You can delete a specified file on the device as shown by the following example:

```cfml
<cfset removeresponse=cfclient.file.remove("myNewFile.txt")>
```

Method summary

`remove(fileName) - deletes file specified by filename`

- **returns** - true for successful removal
- **params**
  - filename – Absolute or relative path of the file to be deleted (including the file name)
**cfclient.file.removeDirectory**

*Deleting a directory*

You can delete a directory by using the `removeDirectory` function as shown in the following example:

```coldfusion
<cfset removeresponse = cfclient.file.removeDirectory("myNewDir")>
```

**Method summary**

```
removeDirectory(dirPath, recurse) - Delete specified directory
```

- **returns** – a boolean to indicate the success of operation
- **params**
  - `dirPath` – The absolute or relative path to the directory that needs to be deleted (including the directory name).
  - `recurse` - boolean to indicate whether the directory should be removed recursively.
**cfclient.file.renameDirectory**

*Renaming a directory*

You can rename a directory using the renameDirectory function as shown in the following example:

```cf
<cfset dirobj = cfclient.file.renameDirectory(oldName,newName)>
```

**Method summary**

renameDirectory(oldName,newName) - renames the directory with newName

- returns – A `DirectoryEntry` object
- params
  - `oldName` – The name of directory with absolute or relative path to be renamed
  - `newName` – The new name for the directory with absolute path

For instance,

```cf
<cfset cfclient.file.renameDirectory('first','second')>
```
**cfclient.file.setFileSystem**

*Setting the file system*

You can set the underlying filesystem for a persistent or temporary storage depending upon how long you want to store the files.

To set the device file system, use the setFileSystem() function.

**Method summary**

`setFileSystem(fileSystem)`

- **returns** – None
- **params** - String (PERSISTENT/TEMPORARY)

- **PERSISTENT** – Permanent storage. Used for storage that should not be removed by the user agent without application or user permission.
- **TEMPORARY** – Temporary storage. Used for storage with no guarantee of persistence

**Example**

```cfc
<cfclientsettings enableDeviceAPI=true>
<cfclient>

  <cfset #cfclient.file.setFileSystem('PERSISTENT')#>

</cfclient>
```
cfclient.file.setWorkingDirectory

Setting the current directory

You can set the current working directory by using the setWorkingDirectory() function.

Method summary

setWorkingDirectory() – Set the current working directory

- **returns** – None
- **params**
  - dirPath – Absolute or relative path of the directory

For instance,

```<cfset cfclient.file.setWorkingDirectory(dirPath)>```
cfclient.file.upload

Upload a file

You can upload a file to a remote server using an HTTP multi-part POST request.

Method summary

cfclient.file.upload(fileName, url, successCallback, errorCallback, [options])

- returns - None
- params
  - fileName – Absolute or relative path of a file name (including file name)
  - url - The remote server URL
  - successCallback - The function to be invoked upon success/completion
  - errorCallback - The function to be invoked if there is a file transfer error
  - options - See this document for more information

Options object

Apart from the options mentioned in this document, the following options are also supported:

- trustAllHosts - Defaults to false. If set to true, it accepts all the security certificates. This is useful because Android rejects self-signed certificates. It is not recommended to set this as true for production use.

Example

See Getting Started Examples

```
//A function to capture and upload an image to a remote server
function uploadImageToServer(imageURI)
{
    var uploadOptions = new FileUploadOptions();
    uploadOptions.fileKey="uploadedImageFile";
    uploadOptions.fileName=imageURI.substr(imageURI.lastIndexOf('/')+1);
    uploadOptions.mimeType="image/jpeg";
    uploadOptions.chunkedMode = false;

    var fileTransferObj = new FileTransfer();

    fileTransferObj.upload(imageURI,
        function(){
            alert("File uploaded to the server");
        },
        function (e){
            alert("Error uploading file to the server");
        }, uploadOptions);
}
```
**cfclient.file.write**

*Writing to a file*

You can write text to a file using the write function. This function creates a file if a given file name does not exist. See the following example:

```cfset cfclient.file.write("myNewFile.txt", "Hello World!!")>
```

**Method summary**

`write(name, data) - write data to a given file`

- **returns** - None
- **params**
  - name – Absolute or relative path of the file name (including file name)
  - data - data to be written to the file
**DirectoryEntry Object**

The DirectoryEntry object represents a directory on a file system of the mobile device.

The properties of DirectoryEntry object are:

- **isFile**: Always false *(boolean)*.
- **isDirectory**: Always true *(boolean)*.
- **name**: The name of the directory, excluding the path leading to it.
- **fullPath**: The full absolute path from the root to the directory.
FileEntry Object

FileEntry object represents a file on a file system of the mobile device.

The properties of FileEntry object are:

- **isFile**: Always true (boolean).
- **isDirectory**: Always false (boolean).
- **name**: The name of the file, excluding the path leading to it.
- **fullPath**: The full absolute path from the root to the file.
rename Function

Renaming a file

You can rename a file using the rename function as shown in the following example:

```
<cfset fileobj = cfclient.file.rename(oldName,newName)>
```

Method summary

rename(oldName,newName) - renames the file with newName

- returns – A FileEntry object
- params
  - oldName – The name of file (absolute/relative path) to be renamed
  - newName – The new name for the file with absolute/relative path
Geolocation Functions

The Geolocation APIs allow your mobile application to connect and get details from the location sensor.

- `cfclient.geolocation.clearWatch`
- `cfclient.geolocation.getCurrentPosition`
- `cfclient.geolocation.getOptions`
- `cfclient.geolocation.setOptions`
- `cfclient.geolocation.watchPosition`
**cfclient.geolocation.clearWatch**

*Stop watching location changes*

You can stop watching for location changes using the `clearWatch()` function.

**Method summary**

`clearWatch(watchid)`

- **returns** – None
- **params**
  - `watchID` (String)

For instance,

```coldfusion
<cfset cfclient.geolocation.clearWatch(watchID)>
```

**Example**

See [Using the Geolocation APIs](#)
cfclient.geolocation.getCurrentPosition

Getting current position

You can get the device's current position using the getCurrentPosition() function.

Method summary

gCurrentPosition(options)

- **returns** – Returns a Position object containing the location information
- **params**
  - Geolocation Options

For instance,

```
<cfset curpos = cfclient.geolocation.getCurrentPosition()>
```

Example

See Using the Geolocation APIs
**cfclient.geolocation.getOptions**

*Getting Geolocation options*

You can get the current Geolocation options using the `getOptions()` function.

**Method summary**

`getOptions()`

- **returns** – An object representing the `Options` object
- **params**
  - None

For instance,

```cfs
<cfscript>
  opt=cfclient.geolocation.getOptions();
  /*Populating the content of the <div> with the ID divId*/
  document.getElementById('divId').innerHTML=" Enable high accuracy: "+opt.enableHighAccuracy;
  document.getElementById('divId').innerHTML=" Maximum age: "+opt.maximumAge;
</cfscript>
```

**Geolocation Options structure**

The Geolocation Options object contains the following parameters:

- **enableHighAccuracy (Boolean)** - By default, the device attempts to retrieve a position using the network-based methods. Setting this property to `true` enables ColdFusion to use more accurate methods, such as satellite positioning.
- **timeout (Number)** – The maximum time length (milliseconds) that is allowed to pass from the call to `getCurrentPosition` or `watchPosition` until the corresponding geolocationSuccess callback gets executed.
- **maximumAge (Number)** - Accept a cached position whose age is no greater than the specified time in milliseconds.

**Example**

See [Using the Geolocation APIs](#)
**cfclient.geolocation.setOptions**

*Setting Geolocation options*

You can set the Geolocation options using the setOptions() function.

**Method summary**

*setOptions(options)*

- **returns** – None
- **params**
  - An object representing the [Geolocations options](#)

For instance,

```
<cfset opt. enableHighAccuracy=false>
<cfset opt. maximumAge=2000>
<cfset opt. timeOut=4000>
<cfset cfclient.geolocation.setOptions(opt)>
```

**Geolocation Options structure**

The Geolocation Options object contains the following parameters:

- **enableHighAccuracy(Boolean)** - By default, the device attempts to retrieve a position using the network-based methods. Setting this property to `true` enables ColdFusion to use more accurate methods, such as satellite positioning.
- **timeOut (Number)** – The maximum time length (milliseconds) that is allowed to pass from the call to `getCurrentPosition` or `watchPosition` until the corresponding `geolocationSuccess` callback gets executed.
- **maximumAge (Number)** - Accept a cached position whose age is no greater than the specified time in milliseconds.

**Example**

See [Using the Geolocation APIs](#)
**cfclient.geolocation.watchPosition**

*Watching for change in location*

Using the `watchPosition` function, you can constantly watch and get notification whenever the location of the device changes.

**Method summary**

`watchPosition(callback, options)`

- **returns** – watchID (String)
- **params**
  - callback (function), options (object)

For instance,

```cfscript
opt=cfclient.geolocation.getOptions();
watchId=cfclient.geolocation.watchPosition(wtchposcallback,opt);
/*Populating the content of the <div> with the ID divId*/
document.getElementById('divId').innerHTML="Watch Id: "+watchId;

/*Populating the content of the <div> with the ID divId*/
document.getElementById('divId').innerHTML="Watch Id: "+watchId;
```

**Note:** We have used alert() in the above example. alert() may not work on iOS devices.

**Example**

See [Using the Geolocation APIs](#)
Media and Capture Functions

ColdFusion enables you to build mobile applications capable of capturing audio and video. APIs are made available for playback and control of audio files.

- Audio Functions
- Video Functions
## Audio Functions

### Supported audio Functions

- [http://docs.phonegap.com/en/2.5.0/cordova_media_capture_capture.md.html#CaptureAudioOptions](http://docs.phonegap.com/en/2.5.0/cordova_media_capture_capture.md.html#CaptureAudioOptions)

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
<th>Returns</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>capture([options], [callback])</td>
<td>Record audio based on provided options</td>
<td>Array of MediaFile object or null based on whether callback function is provided or not</td>
<td>options (Audio Options, callback) The audio options are listed here.</td>
</tr>
<tr>
<td>createMedia(src, [statusCallback])</td>
<td>Create media object for playback</td>
<td>Media Object</td>
<td>src fullpath of media file statusCallback function to get Media Playback status events</td>
</tr>
<tr>
<td>getCurrentPosition(media)</td>
<td>Get current position for given media object</td>
<td>position (Number)</td>
<td>Media Object</td>
</tr>
<tr>
<td>play(src)</td>
<td>Start or resume given audio file</td>
<td>Media object</td>
<td>fullpath or Media Object</td>
</tr>
<tr>
<td>pause(media)</td>
<td>Pause given audio file</td>
<td>void</td>
<td>Media Object</td>
</tr>
<tr>
<td>release(media)</td>
<td>Release underlying audio resources</td>
<td>void</td>
<td>Media Object</td>
</tr>
<tr>
<td>seekTo(media, duration)</td>
<td>Set current position within audio file</td>
<td>void</td>
<td>Media Object duration in milliseconds</td>
</tr>
<tr>
<td>stop(media)</td>
<td>Stop playing given audio file</td>
<td>void</td>
<td>Media Object</td>
</tr>
</tbody>
</table>
### Recording audio

```cfc
<!--- using callback function --->
<cfset cfclient.audio.capture({},recordingComplete)>

<cffunction name="recordingComplete">
  <cfargument name="mediafile"/>
  <!---Populating the content of the <div> with the ID divId--->
  <cfset document.getElementById('divId').innerHTML = mediafile[1].fullPath>
  <cfset document.getElementById('divId').innerHTML = mediafile.size>
</cffunction>

<!--- without callback function --->
<cfset mediafile = cfclient.audio.capture()>
<cfset document.getElementById('divId').innerHTML = mediafile[1].fullPath>
<cfset document.getElementById('divId').innerHTML = mediafile.size>
```

**Note:** The duration parameter of audio options is not supported on Android. The Mode parameter is not supported on both platforms.

### Playing audio

```cfc
<!--- playing file from a given source --->
<cfset media = cfclient.audio.createMedia("/mnt/sdcard/myaudiofile.wav")>
<cfset cfclient.audio.play(media)>
<cfset cfclient.audio.pause(media)>
<cfset cfclient.audio.stop(media)>
<cfset cfclient.audio.release(media)>
```

**More examples**

See [Using the Audio capture APIs](#).
Video Functions

Supported video APIs

<table>
<thead>
<tr>
<th>API name</th>
<th>Description</th>
<th>Return Type</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>capture([options], [callback])</td>
<td>Record video based on provided options</td>
<td>Array of MediaFile object or null based on whether callback function is provided or not</td>
<td>options (Video Options, callBack) The video options are listed here.</td>
</tr>
</tbody>
</table>

For capturing video,

```cf
<cfset opt = cfclient.video.getOptions()>
<cfset med = cfclient.video.capture(opt, recordingComplete)>
```

Example

Using the Video capture APIs
Notification Functions

Visual, audible and tactile device notification support

- `cfclient.notification.alert`
- `cfclient.notification.beep`
- `cfclient.notification.confirm`
- `cfclient.notification.vibrate`
cfclient.notification.alert

Displaying alert messages

Show an alert dialog

Method summary

<table>
<thead>
<tr>
<th>alert (message, title, buttonLabels)</th>
</tr>
</thead>
</table>

- **returns** – None
- **params**
  - message (String)
  - title (String)
  - buttonLabels (String)

For instance,

```cfc
cfset cfclient.notification.alert("Alert!","Notification","Done")
```
**cfclient.notification.beep**

*Creating a beep sound*

beep

**Method summary**

<table>
<thead>
<tr>
<th>beep (count)</th>
</tr>
</thead>
</table>

- **returns** – None
- **params**
  - count - Number of times to repeat the beep sound.

For instance,

```cfset cfclient.notification.beep(2)```
**cfclient.notification.confirm**

*Showing a confirm alert dialog*

Shows a confirm alert dialog

**Method summary**

| confirm (message, title, buttonLabels) |

- **returns** – None
- **params**
  - message (String) – The message on the dialog
  - title (String) – The title of the dialog
  - buttonLabels (String) - Comma-separated string specifying button labels. (String) (Optional, defaults to OK,Cancel)

For instance,

```cf
cfset cfclient.notification.confirm("Notifying you!!!", "Notification", "Done")
```
**cfclient.notification.vibrate**

*Showing a vibrate alert dialog*

Provides a vibration alert

**Method summary**

<table>
<thead>
<tr>
<th>vibrate (milliseconds)</th>
</tr>
</thead>
</table>

- **returns** – None
- **params**
  - milliseconds - The number of milliseconds to vibrate the device.

For instance,

```cfset cfclient.notification.vibrate(1000)>```
Splash Screen Functions

You can show or hide a splash screen image that you have configured as part of your PhoneGap build using the show() and hide() functions.

- `cfclient.splashscreen.hide`
- `cfclient.splashscreen.show`
**cfclient.splashscreen.hide**

To hide a splash screen on your device, use this function:

```cfset cfclient.splashscreen.hide()```
**cfclient.splashscreen.show**

To show a splash screen in your application, use:

```
<cfset cfclient.splashscreen.show(duration)>
```

The show() function requires the duration in milliseconds.
Storage Functions

ColdFusion supports PhoneGap storage APIs that are based on the [Web storage specification](#). Web storage is an important aspect of any client application running on a browser or a device. This section describes how you can access and manage web storage through ColdFusion.

Support for local storage is provided through key-value pairs.

- `cfclient.localstorage.clear`
- `cfclient.localstorage.getItem`
- `cfclient.localstorage.removeItem`
- `cfclient.localstorage.setItem`
cfclient.localstorage.clear

Clearing the storage

Remove all the items from the local storage.

Method summary

<table>
<thead>
<tr>
<th>clear()</th>
</tr>
</thead>
</table>

- **returns** – None
- **params**
  - None

Usage

```cfc
<cfset item = cfclient.localstorage.clear() >
```

Example

See Using the Storage APIs
cfclient.localstorage.getItem

*Getting an item from storage*

Get an item from the local storage identified by a key.

**Method summary**

<table>
<thead>
<tr>
<th>returns</th>
<th>An item (String)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>params</strong></td>
<td></td>
</tr>
<tr>
<td>Key (String)</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

```
<cfset item = cfclient.localstorage.getItem("myKey") >
```

**Example**

See [Using the Storage APIs](#)
**cfclient.localStorage.removeItem**

*Removing an item from storage*

Remove an item from the local storage identified by a key.

**Method summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>removeItem(key, value)</code></td>
<td>Remove an item from the storage identified by a key.</td>
</tr>
</tbody>
</table>

- **returns** – None
- **params**
  - Key (String)

**Usage**

```coldfusion
<cfset cfclient.localStorage.removeItem("myKey")>
```

**Example**

See [Using the Storage APIs](#)
cfclient.localstorage.setItem

Persisting an item in storage

Set an item in the local storage identified by a key.

Method summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setItem(key, value)</td>
<td></td>
</tr>
</tbody>
</table>

- **returns** – None
- **params**
  - Key (String), Value (String)

Usage

```coldfusion
<cfset cfclient.localstorage.setItem("myKey","myValue") >
```

Example

See Using the Storage APIs