Contents

Chapter 1
Introduction

About functions 11
Using this functions reference 11
Functions reference (alphabetical list) 12

Chapter 2
Aggregate functions

Average 18
Count 19
List 20
Max 21
Min 22
StDev 23
StDevP 25
Sum 26
Variance 27
VarianceP 28

Chapter 3
Date functions

Date 32
Day 32
DayName 33
DayNameJ 34
DayOfWeek 34
DayOfYear 35
Month 35
MonthName 36
MonthNameJ 36
WeekOfYear 37
WeekOfYearFiscal 37
Year 38
YearName 39

Chapter 4
Design functions

DatabaseNames 43
FieldBounds 43
FieldComment 44
FieldIDs 45
FieldNames 45
FieldRepetitions 46
FieldStyle 47
FieldType 48
GetNextSerialValue 49
LayoutIDs 49
LayoutNames 50
LayoutObjectNames 50
RelationInfo 51
ScriptIDs 52
ScriptNames 53
TableIDs 53
TableNames 54
ValueListIDs 54
ValueListItems 55
ValueListNames 55
WindowNames 56

Chapter 5
External functions 57

External 58

Chapter 6
Financial functions 59

FV 60
NPV 60
PMT 61
PV 62

Chapter 7
Get functions 63

Get(AccountName) 68
Get(ActiveFieldContents) 68
Get(ActiveFieldName) 69
Get(ActiveFieldTableName) 69
Get(ActiveLayoutObjectName) 70
Get(ActiveModifierKeys) 71
Get(ActiveRepetitionNumber) 71
Get(ActiveSelectionSize) 72
Get(ActiveSelectionStart) 72
Get(AllowAbortState) 73
Get(AllowToolbarState)  73
Get(ApplicationLanguage)  74
Get(ApplicationVersion)  75
Get(CalculationRepetitionNumber)  75
Get(CurrentDate)  76
Get(CurrentHostTimestamp)  76
Get(Time)  77
Get(CurrentTimestamp)  78
Get(CustomMenuSetName)  78
Get(DesktopPath)  79
Get(DocumentsPath)  80
Get(ErrorCaptureState)  80
Get(ExtendedPrivileges)  81
Get(FileMakerPath)  81
Get(File)  82
Get(FilePath)  82
Get(FileSize)  83
Get(FoundCount)  84
Get(HighContrastColor)  84
Get(HighContrastState)  85
Get(HostIPAddress)  85
Get(HostName)  86
Get(LastError)  87
Get(LastMessageChoice)  93
Get(ODBCErrror)  94
Get(LayoutAccess)  94
Get(LayoutCount)  95
Get(LayoutName)  96
Get(LayoutNumber)  96
Get(LayoutTableName)  97
Get(LayoutViewState)  97
Get(MultiUserState)  98
Get(NetworkProtocol)  99
Get(PageNumber)  99
Get(PortalRowNumber)  100
Get(PreferencesPath)  100
Get(PrinterName)  101
Get(PrivilegeSetName)  102
Get(RecordAccess)  102
Get(RecordID)  103
Get(RecordModificationCount)  104
Get(RecordNumber)  105
Get(RecordOpenCount)  105
Get(RecordOpenState)  106
Get(RequestCount)  107
Get(RequestOmitState)  107
Get(ScreenDepth)  108
Get(ScreenHeight)  108
Get(ScreenWidth)  109
Get(ScriptName)  109
Get(ScriptParameter)  110
Get(ScriptResult)  111
Get(SortState)  112
Get(StatusAreaState)  112
Get(SystemDrive)  113
Get(SystemIPAddress)  114
Get(SystemLanguage)  114
Get(SystemNICAddress)  115
Get(SystemPlatform)  115
Get(SystemVersion)  116
Get(TextRulerVisible)  116
Get(TotalRecordCount)  117
Get(UserCount)  117
Get(UserName)  118
Get(UseSystemFormatsState)  119
Get(WindowContentHeight)  119
Get(WindowContentWidth)  120
Get(WindowDesktopHeight)  120
Get(WindowDesktopWidth)  121
Get(WindowHeight)  122
Get(WindowLeft)  122
Get(WindowMode)  123
Get(WindowName)  123
Get(WindowTop)  124
Get(WindowVisible)  125
Get(WindowWidth)  125
Get(WindowZoomLevel)  126

Chapter 8
Logical functions  127
Case  128
Choose  128
Evaluate  129
EvaluationError  131
GetAsBoolean  131  
GetField  132  
GetLayoutObjectAttribute  133  
GetNthRecord  134  
If  135  
IsEmpty  136  
IsValid  137  
IsValidExpression  137  
Let  138  
Lookup  140  
LookupNext  141

Chapter 9  
**Number functions**  143  
Abs  144  
Ceiling  144  
Combination  145  
Div  145  
Exp  146  
Factorial  146  
Floor  147  
Int  147  
Lg  148  
Ln  149  
Log  149  
Mod  150  
Random  150  
Round  151  
SetPrecision  151  
Sign  152  
Sqrt  153  
Truncate  153

Chapter 10  
**Repeating functions**  155  
Extend  156  
GetRepetition  156  
Last  157

Chapter 11  
**Summary functions**  159  
GetSummary  160
Chapter 12

Text functions

Exact  161
Filter  163
FilterValues  164
GetAsCSS  165
GetAsDate  166
GetAsNumber  166
GetAsSVG  167
GetAsText  168
GetAsTime  168
GetAsTimestamp  169
GetAsURLEncoded  169
GetValue  170
Hiragana  171
KanaHankaku  171
KanaZenkaku  172
KanjiNumeral  172
Katakana  172
Left  173
LeftValues  173
LeftWords  174
Length  175
Lower  175
Middle  176
MiddleValues  176
MiddleWords  177
NumToJText  178
PatternCount  179
Position  179
Proper  180
Quote  181
Replace  181
Right  182
RightValues  182
RightWords  183
RomanHankaku  184
RomanZenkaku  184
SerialIncrement  185
Substitute  186
Trim  186
TrimAll  187
Chapter 1
Introduction

About functions

A function is a predefined, named formula that performs a specific calculation and returns a single, specific value.

Most functions include three basic parts:

- the function
- a set of parentheses, if the function takes parameters
- the parameters required by the function

Each function returns a result of field type text, number, date, time, timestamp, or container.

FileMaker® Pro provides many functions for you to use in your FileMaker database solutions.

Using this functions reference

The content in this document was originally written for the FileMaker Pro and FileMaker Pro Advanced Help. It has been collected in this format to allow solution developers to read the information independent of the help system.

The following chapters present the functions by category. FileMaker Pro functions are grouped by the type of data they operate on, not by the type of data they return. For example, the Position function returns a number, but it is grouped with Text functions because it operates on text data.
Functions reference (alphabetical list)

This section lists the functions in alphabetical order.

A, B, C
Abs  144
Atan  208
Average  18
Case  128
Ceiling  144
Choose  128
Combination  145
Cos  208
Count  19

D
DatabaseNames  43
Date  32
Day  32
DayName  33
DayNameJ  34
DayOfWeek  34
DayOfYear  35
Degrees  209
Div  145

E
Evaluate  129
EvaluationError  131
Exact  163
Exp  146
Extend  156
External  58

F
Factorial  146
FieldBounds  43
FieldComment  44
FieldIDs  45
FieldNames  45
FieldRepetitions  46
FieldStyle  47
FieldType  48
Filter  164
FilterValues  164
Floor  147
FV  60

G
Get(AccountName)  68
Get(ActiveFieldContents)  68
Get(ActiveFieldName)  69
Get(ActiveFieldTableName)  69
Get(ActiveLayoutObjectName)  70
Get(ActiveModifierKeys)  71
Get(ActiveRepetitionNumber)  71
Get(ActiveSelectionSize)  72
Get(ActiveSelectionStart)  72
Get(AllowAbortState)  73
Get(AllowToolbarState)  73
Get(ApplicationLanguage)  74
Get(ApplicationVersion)  75
Get(CalculationRepetitionNumber)  75
Get(CurrentDate)  76
Get(CurrentHostTimestamp)  76
Get(CurrentTime)  77
Get(CurrentTimestamp)  78
Get(CustomMenuSetName)  78
Get(DesktopPath)  79
Get(DocumentsPath)  80
Get(ErrorCaptureState)  80
Get(ExtendedPrivileges)  81
Get(FileMakerPath)  81
Get(FileName)  82
Get(FilePath)  82
Get(FileSize)  83
Get(FoundCount)  84
Get(HighContrastColor)  84
Get(HighContrastState)  85
Get(HostIPAddress)  85
Get(HostName)  86
Get(LastError)  87
Get(LastMessageChoice)  93
Get(LastODBCError)  94
Get(LayoutAccess)  94
Get(LayoutCount)  95
Get(LayoutName)  96
Get(LayoutNumber)  96
Get(LayoutTableName)  97
Get(LayoutViewState)  97
Get(MultiUserState)  98
Get(NetworkProtocol)  99
Get(PageNumber)  99
Get(PortalRowNumber)  100
Get(PreferencesPath)  100
Get(PrinterName)  101
Get(PrivilegeSetName)  102
Get(RecordAccess)  102
Get(RecordID)  103
Get(RecordModificationCount)  104
Get(RecordNumber)  105
Get(RecordOpenCount)  105
Get(RecordOpenState)  106
Get(RequestCount)  107
Get(RequestOmitState)  107
Get(ScreenDepth)  108
Get(ScreenHeight)  108
Get(ScreenWidth)  109
Get(ScriptName)  109
Get(ScriptParameter)  110
Get(ScriptResult)  111
Get(SortState)  112
Get(StatusAreaState)  112
Get(SystemDrive)  113
Get(SystemIPAddress)  114
Get(SystemLanguage)  114
Get(SystemNICAddress)  115
Get(SystemPlatform)  115
Get(SystemVersion)  116
Get(TextRulerVisible)  116
Get(TotalRecordCount)  117
Get(UserCount)  117
Get(UserName)  118
Get(UseSystemFormatsState)  119
Get(WindowContentHeight)  119
Get(WindowContentWidth)  120
Get(WindowDesktopHeight)  120
Get(WindowDesktopWidth)  121
Get(WindowHeight)  122
Get(WindowLeft)  122
Get(WindowMode)  123
Get(WindowName)  123
Get(WindowTop)  124
Get(WindowVisible)  125
Get(WindowWidth)  125
Get(WindowZoomLevel)  126
GetAsBoolean  131
GetAsCSS  165
GetAsDate  166
GetAsNumber  166
GetAsSVG  167
GetAsText  168
GetAsTime  168
GetAsTimestamp  169
GetAsURLEncoded  169
GetField  132
GetLayoutObjectAttribute  133
GetNextSerialValue  49
GetNthRecord  156
GetRepetition  156
GetSummary  160
GetValue  170

H, I, J, K
Hiragana  171
Hour  202
If  135
Int  147
IsEmpty  136
IsValid  137
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsValidExpression</td>
<td>137</td>
</tr>
<tr>
<td>KanaHankaku</td>
<td>171</td>
</tr>
<tr>
<td>KanaZenkaku</td>
<td>172</td>
</tr>
<tr>
<td>KanjiNumeral</td>
<td>172</td>
</tr>
<tr>
<td>Katakana</td>
<td>172</td>
</tr>
<tr>
<td><strong>L, M, N, O</strong></td>
<td></td>
</tr>
<tr>
<td>Last</td>
<td>157</td>
</tr>
<tr>
<td>LayoutIDs</td>
<td>49</td>
</tr>
<tr>
<td>LayoutNames</td>
<td>50</td>
</tr>
<tr>
<td>LayoutObjectNames</td>
<td>50</td>
</tr>
<tr>
<td>Left</td>
<td>173</td>
</tr>
<tr>
<td>LeftValues</td>
<td>173</td>
</tr>
<tr>
<td>LeftWords</td>
<td>174</td>
</tr>
<tr>
<td>Length</td>
<td>175</td>
</tr>
<tr>
<td>Let</td>
<td>138</td>
</tr>
<tr>
<td>Lg</td>
<td>148</td>
</tr>
<tr>
<td>List</td>
<td>20</td>
</tr>
<tr>
<td>Ln</td>
<td>149</td>
</tr>
<tr>
<td>Log</td>
<td>149</td>
</tr>
<tr>
<td>Lookup</td>
<td>140</td>
</tr>
<tr>
<td>LookupNext</td>
<td>141</td>
</tr>
<tr>
<td>Lower</td>
<td>175</td>
</tr>
<tr>
<td>Max</td>
<td>21</td>
</tr>
<tr>
<td>Middle</td>
<td>176</td>
</tr>
<tr>
<td>MiddleValues</td>
<td>176</td>
</tr>
<tr>
<td>MiddleWords</td>
<td>177</td>
</tr>
<tr>
<td>Min</td>
<td>22</td>
</tr>
<tr>
<td>Minute</td>
<td>202</td>
</tr>
<tr>
<td>Mod</td>
<td>150</td>
</tr>
<tr>
<td>Month</td>
<td>35</td>
</tr>
<tr>
<td>MonthName</td>
<td>36</td>
</tr>
<tr>
<td>MonthNameJ</td>
<td>36</td>
</tr>
<tr>
<td>NPV</td>
<td>60</td>
</tr>
<tr>
<td>NumToJText</td>
<td>178</td>
</tr>
<tr>
<td><strong>P, Q</strong></td>
<td></td>
</tr>
<tr>
<td>PatternCount</td>
<td>179</td>
</tr>
<tr>
<td>Pi</td>
<td>209</td>
</tr>
<tr>
<td>PMT</td>
<td>61</td>
</tr>
<tr>
<td>Position</td>
<td>179</td>
</tr>
<tr>
<td>Proper</td>
<td>180</td>
</tr>
<tr>
<td>PV</td>
<td>62</td>
</tr>
<tr>
<td>Quote</td>
<td>181</td>
</tr>
<tr>
<td><strong>R</strong></td>
<td></td>
</tr>
<tr>
<td>Radians</td>
<td>210</td>
</tr>
<tr>
<td>Random</td>
<td>150</td>
</tr>
<tr>
<td>RelationInfo</td>
<td>51</td>
</tr>
<tr>
<td>Replace</td>
<td>181</td>
</tr>
<tr>
<td>RGB</td>
<td>192</td>
</tr>
<tr>
<td>Right</td>
<td>182</td>
</tr>
<tr>
<td>RightValues</td>
<td>182</td>
</tr>
<tr>
<td>RightWords</td>
<td>183</td>
</tr>
</tbody>
</table>
RomanHankaku 184
RomanZenkaku 184
Round 151

S
ScriptIDs 52
ScriptNames 53
Seconds 203
SerialIncrement 185
SetPrecision 151
Sign 152
Sin 210
Sqrt 153
StDev 23
StDevP 25
Substitute 186
Sum 26

T, U
TableIDs 53
TableNames 54
Tan 211
TextColor 192
TextColorRemove 193
TextFont 194
TextFontRemove 195
TextFormatRemove 196
TextSize 197
TextSizeRemove 197
TextStyleAdd 198
TextStyleRemove 199
Time 203
Timestamp 206
Trim 186
TrimAll 187
Truncate 153
Upper 188

V, W, X, Y, Z
ValueCount 189
ValueListIDs 54
ValueListItem 55
ValueListNames 55
Variance 27
VarianceP 28
WeekOfYear 37
WeekOfYearFiscal 37
WindowNames 56
WordCount 189
Year 38
YearName 39
Chapter 2
Aggregate functions

Aggregate functions perform statistical analysis on numbers (and also dates or times for some functions) in:

- several fields in a record.
- related fields whether displayed in a portal or not.
- repeating fields.

For example, you can use the `Sum` function to add the values listed in a portal, as an alternative to creating a report with grouped data and subtotals.

The parameter values can include a numeric constant (for example, 10) or any valid expression. A constant parameter in a formula for a repeating field affects the result for every repetition.

When repeating field parameters (field1; field2;...) include a non-repeating field, that value is used in the result for only the first repetition unless you use the Extend function, page 156.

Values in repetitions that exceed the number of repetitions in the calculated field are ignored. For example, a calculated field with three repetitions holds only three results, even when one field referenced in the calculation has five repetitions.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average, page 18</td>
<td>The average of all valid, non-blank values in the specified field.</td>
</tr>
<tr>
<td>Count, page 19</td>
<td>The number of valid, non-blank values in the specified field.</td>
</tr>
<tr>
<td>List, page 20</td>
<td>The concatenation of all non-blank values in list form, separated by carriage returns.</td>
</tr>
<tr>
<td>Max, page 21</td>
<td>The highest valid value in a field or fields.</td>
</tr>
<tr>
<td>Min, page 22</td>
<td>The smallest valid non-blank value in a field or fields.</td>
</tr>
<tr>
<td>StDev, page 23</td>
<td>The standard deviation of a series of valid non-blank values in a field or fields.</td>
</tr>
<tr>
<td>StDevP, page 25</td>
<td>The standard deviation of a population represented by a series of valid non-blank values in a field or fields.</td>
</tr>
<tr>
<td>Sum, page 26</td>
<td>The total of all valid, non-blank values in the specified fields.</td>
</tr>
<tr>
<td>Variance, page 27</td>
<td>The variance of a series of valid non-blank values in a field or fields.</td>
</tr>
<tr>
<td>VarianceP, page 28</td>
<td>The variance of a population in a series of valid non-blank values in a field or fields.</td>
</tr>
</tbody>
</table>
**Average**

**Format**

Average(field{;field...})

**Parameter**

field - any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields.

Parameters in curly braces {} are optional.

**Data type returned**

number

**Description**

Returns a value that is the average of all valid, non-blank values in field, where field can be any of the following:

- a repeating field (repeatingField).
- a field in matching related records specified by (table::field), whether or not these records appear in a portal.
- several non-repeating fields in a record (field1;field2;field3...).
- corresponding repetitions of repeating fields in a record (repeatingField1;repeatingField2;repeatingField3), if the result is returned in a repeating field with at least the same number of repeats.
- several fields in the first matching record specified by (table::field1;table::field2;...). You can include fields from different tables (table 1::field A;table 2::field B...).

**Examples**

A Student table has a portal showing scores for all exams a student has taken. The exam scores are in a table called Exams.

Average(Exams::Score) returns the student’s average score for all exams she has taken.

In the following examples:

- Field1 contains two repetitions with values of 1 and 2.
- Field2 contains four repetitions with values of 5, 6, 7, and 8.
- Field3 contains 6.

Average(Field2) returns 6.5 when the calculation isn’t a repeating field.

Average(Field1;Field2;Field3) returns 4, 4, 7, 8 when the calculation is a repeating field.
Note When a referenced field is a repeating field, the Average function returns the average of the values in the first repetition field, then the average of the values in the second repetition field, and so on. Therefore, \((1+5+6)/3=4;(2+6)/2=4;7/1=7;8/1=8\).

**Count**

**Format**

Count(field{;field...})

**Parameter**

field - any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields.

Parameters in curly braces {} are optional.

**Data type returned**

number

**Description**

Returns the number of valid, non-blank values in field where field can be any of the following:

- a repeating field (repeatingField).
- a field in matching related records specified by (table::field), whether or not these records appear in a portal.
- several non-repeating fields in a record (field1;field2;field3...).
- corresponding repetitions of repeating fields in a record (repeatingField1;repeatingField2;repeatingField3), if the result is returned in a repeating field with at least the same number of repeats.
- several fields in the first matching record specified by (table::field1;table::field2;...). You can include fields from different tables (table 1::field A;table 2::field B...).

**Examples**

The Accounts layout has a portal showing installment payments made.

Count(Payments::Payment) returns the number of payments made on an account.

In the following examples:

- Field1 contains two repetitions with values of 1 and 2.
- Field2 contains four repetitions with values of 5, 6, 7, and 8.
- Field3 contains 6.

Count(Field2) returns 4 when the calculation isn’t a repeating field.
Count(Field1;Field2;Field3) returns 3, 2, 1,1 when the calculation is a repeating field.

**Note** When a referenced field is a repeating field, the Count function returns the total number of valid, non-blank values in the first repetition field, then the number of valid, non-blank values in the second repetition field, and so on.

### List

**Format**

List(field{;field...})

**Parameter**

field - any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields.

Parameters in curly braces {} are optional.

**Data type returned**

text

**Description**

Returns a concatenated list of non-blank values (separated by carriage returns) for either:

- a single field (table::field), calculates a single result over all repetitions (if any) for this field and over all matching related records, whether or not these records appear in a portal.
- several fields and/or literal values (table::field1,constant,table::field2...), calculates a separate result for each repetition of the calculation across each corresponding repetition of the fields. If any fields are related, only the first related record is used.

**Examples**

In the following examples:

- Field1 contains white.
- Field2 contains black.
- Field3 contains three repetitions with values of red, green, blue.
- Related::Field4 refers to three records that contain 100, 200, 300.

**Note** When referencing multiple repeating fields, List() returns the list of the values across the first repetition in the calculation's first repetition, then the list of the values across the second repetition in the second repetition, and so on.
Example 1
List (Field1; Field2) returns:
white
black
Example 2
List(Field3) returns:
red
green
blue
Example 3
List (Field1; Field2; Field3) returns:
in calculation repetition 1:
white
black
red
in calculation repetition 2:
green
in calculation repetition 3:
blue
Example 4
List(Related::Field4) returns:
100
200
300

Max

Format
Max(field{;field...})

Parameter
field - any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields.

Parameters in curly braces {} are optional.
Data type returned
text, number, date, time, timestamp

Description
Returns the highest valid value in:
• a repeating field (repeatingField).
• a field in matching related records specified by (table::field), whether or not these records appear in a portal.
• several non-repeating fields in a record (field1;field2;field3...).
• corresponding repetitions of repeating fields in a record (repeatingField1;repeatingField2;repeatingField3), if the result is returned in a repeating field with at least the same number of repeats.
• several fields in the first matching record specified by (table::field1;table::field2;...). You can include fields from different tables (table 1::field A;table 2::field B...).

Examples
The Accounts layout has a portal showing installment payments made.
Max(Payments::PaymentDate) returns the most recent date a payment was made on an account.
In the following examples:
• Field1 contains two repetitions with values of 1 and 2.
• Field2 contains four repetitions with values of 5, 6, 7, and 8.
• Field3 contains 6.
Max(Field2) returns 8 when the calculation isn’t a repeating field.
Max(Field1;Field2;Field3) returns 6, 6, 7, 8 when the calculation is a repeating field.
Note When a referenced field is a repeating field, the Max function returns the maximum value in the first repetition field, then the maximum value in the second repetition field, and so on.

Min

Format
Min(field{;field...})

Parameter
field - any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields.
Parameters in curly braces { } are optional.

**Data type returned**

- text, number, date, time, timestamp

**Description**

Returns the smallest valid non-blank value in:

- a repeating field (repeatingField).
- a field in matching related records specified by (table::field), whether or not these records appear in a portal.
- several non-repeating fields in a record (field1;field2;field3...).
- corresponding repetitions of repeating fields in a record (repeatingField1;repeatingField2;repeatingField3), if the result is returned in a repeating field with at least the same number of repeats.
- several fields in the first matching record specified by (table::field1;table::field2;...). You can include fields from different tables (table 1::field A;table 2::field B...).

**Examples**

A Contracts table has a portal showing bids submitted for each contract.

Min(Bids::Price) returns the lowest bid submitted for a contract.

In the following examples:

- Field1 contains two repetitions with values of 1 and 2.
- Field2 contains four repetitions with values of 5, 6, 7, and 8.
- Field3 contains 6.

Min(Field2) returns 5 when the calculation isn’t a repeating field.

Min(Field1;Field2;Field3) returns 1, 2, 7, 8 when the calculation is a repeating field.

**Note** When a referenced field is a repeating field, the Min function returns the minimum value in the first repetition field, then the minimum value in the second repetition field, and so on.

**StDev**

**Format**

StDev(field{;field...})
Parameter

field - any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields.

Parameters in curly braces {} are optional.

Data type returned

number

Description

Returns the standard deviation of the sample represented by a series of non-blank values in:

- a repeating field (repeatingField).
- a field in matching related records specified by (table::field), whether or not these records appear in a portal.
- several non-repeating fields in a record (field1;field2;field3).
- corresponding repetitions of repeating fields in a record (repeatingField1;repeatingField2;repeatingField3), if the result is returned in a repeating field with at least the same number of repeats.
- several fields in the first matching record specified by (table 1::field A, table 2::field B,...). You can name a different table for each field (table 1::field A;table 2::field B...).

\[
\text{StDev} = \sqrt{\frac{x_1^2 + x_2^2 + \ldots + x_n^2}{n-1} - \left(\frac{x_1 + x_2 + \ldots + x_n}{n}\right)^2}
\]

Examples

A portal displays the related values 5, 6, 7, and 8 in a field called Scores.

\text{StDev}(table::Scores) returns 1.29099444....

In the following examples:

- Field1 contains two repetitions with values of 1 and 2.
- Field2 contains four repetitions with values of 5, 6, 7, and 8.
- Field3 contains four repetitions with values of 6, 0, 4, and 4.
- Field4 contains one repetition with a value of 3.

\text{StDev(Field4)} results in an error because standard deviation of a single number is not defined.

\text{StDev(Field1;Field2;Field3)} returns 2.64575131..., 3.05505046..., 2.12132034..., 2.82842712... for a repeating field.

Note When a referenced field is a repeating field, the \text{StDev} function returns the standard deviation in the first repetition fields, then the standard deviation in the second repetition fields, and so on.
**StDevP**

**Format**

\[ \text{StDevP}\{\text{field};\text{field}... \} \]

**Parameter**

- **field** - any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields.

Parameters in curly braces {} are optional.

**Data type returned**

number

**Description**

Returns the standard deviation of a population represented by a series of non-blank values in:

- a repeating field \((\text{repeatingField})\).
- a field in matching related records specified by \((\text{table}::\text{field})\), whether or not these records appear in a portal.
- several non-repeating fields in a record \((\text{field1};\text{field2};\text{field3}...\))
- corresponding repetitions of repeating fields in a record \((\text{repeatingField1};\text{repeatingField2};\text{repeatingField3})\), if the result is returned in a repeating field with at least the same number of repeats.
- several fields in the first matching record specified by \((\text{table}::\text{field1};\text{table}::\text{field2};...\)). You can include fields from different tables \((\text{table 1}::\text{field A};\text{table 2}::\text{field B}...\)).

\[
\text{StDevP} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2}{n}}
\]

**Examples**

A portal displays the related values 5, 6, 7, and 8 in the field Scores.

\[ \text{StDevP}(\text{table}::\text{Scores}) \text{ returns 1.11803398...} \]

In the following examples:

- Field1 contains two repetitions with values of 1 and 2.
- Field2 contains four repetitions with values of 5, 6, 7, and 8.
- Field3 contains four repetitions with values of 6, 0, 4, and 4.
- Field4 contains one repetition with a value of 3.

\[ \text{StDevP}(\text{Field4}) \text{ results in an error because the population standard deviation of a single number is not defined.} \]
StDevP(\text{Field2}) \text{ returns } 1.11803398... \text{ for a non-repeating field.}

StDevP(\text{Field1};\text{Field2};\text{Field3}) \text{ returns } 2.16024689..., 2.49443825..., 1.5, 2 \text{ for repeating fields.}

\textbf{Note} \text{ When a referenced field is a repeating field, the StDevP function returns the standard deviation of a population in the first repetition fields, then the standard deviation of a population in the second repetition fields, and so on.}

\textbf{Sum}

\textbf{Format}

\text{Sum(field{};field...)}

\textbf{Parameter}

\text{field} - \text{ any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields. Parameters in curly braces \{\} are optional.}

\textbf{Data type returned}

\text{number}

\textbf{Description}

Returns the total of all valid, non-blank values in:

- a repeating field (\text{repeatingField}).
- a field in matching related records specified by (\text{table::field}), whether or not these records appear in a portal.
- several non-repeating fields in a record (field1;field2;field3...).
- corresponding repetitions of repeating fields in a record (\text{repeatingField1};\text{repeatingField2};\text{repeatingField3}), if the result is returned in a repeating field with at least the same number of repeats.
- several fields in the first matching record specified by (\text{table::field1};\text{table::field2};...). You can include fields from different tables (\text{table 1::field A};\text{table 2::field B}...).

\textbf{Examples}

An Invoice table has a portal showing line items.

\text{Sum(LineItems::ExtendedPrice)} \text{ totals the amounts for all items on the invoice.}

A TimeBilling table has a portal showing time worked on a project. Hours is a time field.

\text{Sum(Hours::BillableHours)} \text{ returns the total number of billable hours on a project. Thus, if the portal shows 40 hours and 15:30 hours, the total billable hours are 55:30, or 55 1/2 hours.}
In the following examples:

- **Field1** contains two repetitions with values of 1 and 2.
- **Field2** contains four repetitions with values of 5, 6, 7, and 8.
- **Field3** contains 6.

If the calculation result isn’t a repeating field:

- \( \text{Sum(Field2)} \) returns 26.
- \( \text{Sum(Field1;Field2;Field3)} \) returns 12.

If the calculation result is a repeating field:

- \( \text{Sum(Field2)} \) returns a repeating field with 26 in the first repetition.
- \( \text{Sum(Field1;Field2;Field3)} \) returns a repeating field with 12, 8, 7, 8.

**Note** When a referenced field is a repeating field, the \( \text{Sum} \) function returns the sum of the first repetition field, then the sum of the second repetition field, and so on.

### Variance

#### Format

\[ \text{Variance(field{;field...})} \]

#### Parameter

- **field** - any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields.

Parameters in curly braces { } are optional.

#### Data type returned

- number

#### Description

Returns the variance of a sample represented by a series of non-blank values. The variance of a distribution is a measure of how spread out the distribution is. Use this function on any of the following fields:

- a repeating field (repeatingField).
- a field in matching related records specified by (table::field), whether or not these records appear in a portal.
- several non-repeating fields in a record (field1;field2;field3...).
- corresponding repetitions of repeating fields in a record (repeatingField1;repeatingField2;repeatingField3), if the result is returned in a repeating field with at least the same number of repeats.
• several fields in the first matching record specified by 
  (table::field1;table::field2;...). You can include fields from different 
  tables (table 1::field A;table 2::field B...).

\[
\text{Variance} = \frac{s_1^2 + s_2^2 + \cdots + s_n^2}{n-1} - \left(\frac{s_1 + s_2 + \cdots + s_n}{n}\right)^2
\]

**Examples**

A portal displays the related values 5, 6, 7, and 8 in Scores.

\[\text{Variance(table::Scores)} \text{ returns } 1.66666666666666...\]

In the following examples:

• Field1 contains two repetitions with values of 1 and 2.
• Field2 contains four repetitions with values of 5, 6, 7, and 8.
• Field3 contains four repetitions with values of 6, 0, 4, and 4.
• Field4 contains one repetition with a value of 3.

\[\text{Variance(Field4)} \text{ results in an error since the variance of a single value is not defined.}\]

\[\text{Variance(Field1;Field2;Field3)} \text{ returns } 7, 9.33333333333333... \]

Student example:

Two classes of students take an exam. Class 1 has scores of 70, 71, 70, 74, 75, 73, 72, and Class 2 has scores of 55, 80, 75, 40, 65, 50, 95. The variance for each class is:

Class 1: \[3.809523809523809\]

Class 2: \[361.9047619047619\]

The variance for Class 1 is much lower than the variance for Class 2, because the scores for Class 2 are more spread out.

**VarianceP**

**Format**

VarianceP(field{;field}...)

**Parameter**

field - any related field, repeating field, or set of non-repeating fields; or an expression that returns a field, repeating field, or set of non-repeating fields.

Parameters in curly braces {} are optional.

**Data type returned**

number
Description

Returns the variance of a population represented by a series of non-blank values. The variance of a population distribution is a measure of how spread out the distribution is. Use this function on any of the following fields:

- a repeating field (repeatingField).
- a field in matching related records specified by (table::field), whether or not these records appear in a portal.
- several non-repeating fields in a record (field1;field2;field3...).
- corresponding repetitions of repeating fields in a record (repeatingField1;repeatingField2;repeatingField3), if the result is returned in a repeating field with at least the same number of repeats.
- several fields in the first matching record specified by (table1::field1;table1::field2;...). You can include fields from different tables (table 1::field A;table 2::field B...).

\[
\text{VarianceP} = \frac{(x_1^2 + x_2^2 + \ldots + x_n^2) - \left(\frac{x_1 + x_2 + \ldots + x_n}{n}\right)^2}{n}
\]

Examples

A portal displays the related values 5, 6, 7, and 8 in Scores.

\(\text{VarianceP}(\text{table::Scores})\) returns 1.25.

In the following examples:

- Field1 contains two repetitions with values of 1 and 2.
- Field2 contains four repetitions with values of 5, 6, 7, and 8.
- Field3 contains four repetitions with values of 6, 0, 4, and 4.
- Field4 contains one repetition with a value of 3.

\(\text{VarianceP}(\text{Field4})\) results in an error since the variance of a single value is not defined.

\(\text{VarianceP}(\text{Field1};\text{Field2};\text{Field3})\) returns 4.66666666..., 6.22222222..., 2.25, 4 if the calculation is a repeating field.

Student example:

Two classes of students take an exam. Class 1 has scores of 70, 71, 70, 74, 75, 73, 72 and Class 2 has scores of 55, 80, 75, 40, 65, 50, 95. The population variance for each class is:

Class 1: 3.26530612...

Class 2: 310.20408163...

The population variance for Class 1 is much lower than the population variance for Class 2 because the scores for Class 1 are more tightly clustered.
Chapter 3

Date functions

Date functions calculate dates and manipulate date information.

**Important** To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

**Note** System formats affect the way dates are displayed. See FileMaker Pro help.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date, page 32</td>
<td>The calendar date for the specified month, day, and year.</td>
</tr>
<tr>
<td>Day, page 32</td>
<td>A number in the range 1 through 31, representing the day of the month for a specified date.</td>
</tr>
<tr>
<td>DayName, page 33</td>
<td>A text string that is the full name of the weekday for the specified date.</td>
</tr>
<tr>
<td>DayNameJ, page 34</td>
<td>A text string that is the full name of the weekday for the specified date in Japanese.</td>
</tr>
<tr>
<td>DayOfWeek, page 34</td>
<td>A number representing the day of the week the specified date falls on.</td>
</tr>
<tr>
<td>DayOfYear, page 35</td>
<td>A number equal to the number of days from the beginning of the year of the specified date.</td>
</tr>
<tr>
<td>Month, page 35</td>
<td>A number in the range 1 through 12, representing the number of the month of the year in which the specified date occurs.</td>
</tr>
<tr>
<td>MonthName, page 36</td>
<td>The name of the month for the specified date.</td>
</tr>
<tr>
<td>MonthNameJ, page 36</td>
<td>The name of the month in Japanese for the specified date.</td>
</tr>
<tr>
<td>WeekOfYear, page 37</td>
<td>The number of weeks after January 1 of the year of the specified date.</td>
</tr>
<tr>
<td>WeekOfYearFiscal, page 37</td>
<td>A number between 1 and 53 representing the week containing a specified date, figured according to the specified starting day.</td>
</tr>
<tr>
<td>Year, page 38</td>
<td>A number representing the year in which the specified date occurs.</td>
</tr>
<tr>
<td>YearName, page 39</td>
<td>The Japanese year name of the specified date, provided in the specified format.</td>
</tr>
</tbody>
</table>
Date

**Format**

Date(month;day;year)

**Parameters**

* month - the month of the year (a number from 1 to 12).
* day - the day of the month (a number from 1 to 31).
* year - the year (four digits between 0001 and 4000. For example, 2005 but not 05).

**Important** The order of the parameters in the Date function is always Month, Day, Year, no matter what operating system or FileMaker Pro date formats you are using.

**Data type returned**

date

**Description**

Returns the calendar date for month, day, and year.

The format of the result depends on the date format that was in use when the database file was created. In the United States, dates are generally in the format MM/DD/YYYY. You can change the date format in the Regional Settings Control Panel (Windows 2000), the Date and Time Control Panel (Windows XP), or the Date & Time System Preference (Mac OS).

You can change how the date is displayed by assigning a different date format to the field in **Layout mode**. Changing the formatting in this way only affects the way the data is displayed, not how it is stored.

**Important** To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

**Note** If you type a month greater than 12 or a day greater than the number of days in a month, FileMaker Pro adds the extra days or months to the result. For example, Date(13;1;2004) returns 1/1/2005.

**Example**


Day

**Format**

Day(date)

**Parameter**

date - any calendar date
**Data type returned**
number

**Description**
Returns a number in the range 1 through 31, representing the day of the month on which date occurs. For example, you can identify the day of the month that payments are due.

**Important** To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

**Examples**
Day(“5/15/2004”) returns 15. This example assumes that the system date format is MM/DD/YYYY.
Day(DateSold) returns the day of the month stored in DateSold.
If(Day(Get(CurrentDate))= 15 and Month(Get(CurrentDate))=3; “Beware the Ides of March”; “”) displays the text Beware the Ides of March only when the day of the month returned by Get(CurrentDate) is 15 and the month returned by Get(CurrentDate) is 3; otherwise it displays nothing.

**DayName**

**Format**
DayName(date)

**Parameter**
date - any calendar date

**Data type returned**
text

**Description**
Returns a text string that is the full name of the weekday for date.

**Important** To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

**Examples**
DayName(Date(10;7;2004)) returns Thursday.
DayName(ProjectDue) returns Friday when ProjectDue is 10/8/2004.
DayName(“10/10/2004”) returns Sunday.

“Return your selection by “& DayName(DueDate) displays the text Return your selection by followed by the name of the day stored in DueDate.
DayNameJ

Format
DayNameJ(date)

Parameter
date - any calendar date

Data type returned
text

Description
Returns a text string in Japanese that is the full name of the weekday for date.

Important To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

Example
DayNameJ(Date(4;4;2003)) returns 金曜日

DayOfWeek

Format
DayOfWeek(date)

Parameter
date - any calendar date

Data type returned
number

Description
Returns a number representing the day of the week that date falls on. The number 1 represents Sunday, 2 represents Monday, 3 represents Tuesday, and so on. For example, you can find out what day of the week a holiday falls on.

Important To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

Examples
DayOfWeek(Date(10;9;2004)) returns 7.
DayOfWeek(ProjectDue) returns 2 when the date in ProjectDue is 10/11/2004.
DayOfYear

**Format**
DayOfYear(date)

**Parameter**
date - any calendar date

**Data type returned**
number

**Description**
Returns a number equal to the number of days from the beginning of the year of date.  
**Important** To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

**Examples**
DayOfYear(Billing Date) returns 33, when Billing Date is 2/2/2005.  
The following formulas return the total number of days in the current year:
DayOfYear(Date(12;31;Year(Get(CurrentDate))))
DayOfYear(Date(1;1;Year(Get(CurrentDate)) + 1) - 1)

Month

**Format**
Month(date)

**Parameter**
date - any calendar date

**Data type returned**
number

**Description**
Returns a number in the range 1 through 12, representing the number of the month of the year in which date occurs.  
**Important** To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.
Examples

Month("3/19/2004") returns 3. This example assumes that the operating system date format is set to MM/DD/YYYY.

Month(Payment) returns 3, where Payment contains March 16, 2004. (The Payment field must be of type date.)

“Bill Due by: “ & Date(Month(DateSold) + 1;Day(DateSold);Year(DateSold)) returns Bill Due by: followed by a value that is one month later than DateSold.

MonthName

Format

MonthName(date)

Parameter

date - any calendar date

Data type returned
text

Description

Returns the full name of the month for date.

Important To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

Examples

MonthName("6/6/2004") returns June.

“Payment due by the end of: “ & MonthName(Date(Month(InvoiceDate) + 1;Day(InvoiceDate);Year(InvoiceDate))) returns Payment due by the end of May, where InvoiceDate is 4/4/2004.

“Payment for: “ & MonthName(Date(Month(Payment) + 1;Day(Payment);Year(Payment))) returns Payment for: followed by the name of the month that is one past the month of the last payment.

MonthNameJ

Format

MonthNameJ(date)

Parameter

date - any calendar date
Data type returned
text

Description
Returns the name of the month of date in Japanese.

Important To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

Example
MonthNameJ("6/6/2003") returns 6 月

WeekOfYear

Format
WeekOfYear(date)

Parameter
date - any calendar date

Data type returned
number

Description
Returns the number of weeks after January 1 of the year of date. Fractions of weeks occurring at the beginning or end of the year count as full weeks, so the WeekOfYear function returns values 1 through 54.

Important To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

Examples
WeekOfYear("1/1/2005") returns 1.
WeekOfYear(ProjectDue) returns 6, when ProjectDue is 2/2/2005.

WeekOfYearFiscal

Format
WeekOfYearFiscal(date;startingDay)

Parameters
date - any calendar date
startingDay - any number between 1 and 7, where 1 represents Sunday
Data type returned
number

Description
Returns a number between 1 and 53 representing the week containing date, figured according to startingDay. startingDay indicates which day is considered the first day of the week.

The first week of the year is the first week that contains four or more days of that year. For example, if you select 1 (Sunday) as the starting day, then January 1 must be on Sunday, Monday, Tuesday, or Wednesday for that week to be the first week of the fiscal year. If you select 2 (Monday) as the starting day, then January 1 must be on Monday, Tuesday, Wednesday, or Thursday for that week to be the first week of the fiscal year.

It is possible, using this function, that dates in a particular year will be returned as the 53rd week of the previous year. For example, if in 2003 you selected Sunday (1) as the starting date, then January 1, 2, or 3 in 2004 would occur in week 53 of fiscal year 2003 (in 2004, January 1 is on a Thursday). The first day of fiscal year 2004 would be on Sunday, January 4, because you selected Sunday (1) as the starting day.

Important To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

Examples
WeekOfYearFiscal(Date(1;3;2004);1) returns 53.
WeekOfYearFiscal(Date(1;4;2004);1) returns 1.
WeekOfYearFiscal(Date(1;1;2004);5) returns 1.

Year

Format
Year(date)

Parameter
date - any calendar date

Data type returned
number

Description
Returns a number representing the year in which date occurs. For example, you can extract the year from a field containing the date an item sold.

Important To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.
Examples

Year(DateSold) returns the year stored in DateSold.
Year(Date(Month(Get(CurrentDate)) + 48; Day(Get(CurrentDate)); Year(Get(CurrentDate)))) returns the year that is 48 months from today’s date.

YearName

Format

YearName(date; format)

Parameters

date - any calendar date
format - a number (0, 1, or 2) that describes the display format

Data type returned
text

Description

Returns the Japanese year name of date, provided in the specified format. If the value for format is blank or other than 0, 1, or 2, then 0 is used.

Format

0 - Meiji (明治) 8, Taisho (大正) 8, Showa (昭和) 8, Heisei (平成) 8 (before 1868.9.8, Seireki (西暦xxxx))
1 - Mei (明) 8, Tai (大) 8, Sho (昭) 8, Hei (平) 8 (before 1868.9.8, Sei (西暦xxxx))
2 - M8, T8, S8, H8 (before 1868.9.8, A.D.xxxx)

Name of Emperor in 0 = Long, 1 = Abbreviated, 2 = 2 byte Roman. Seireki is returned when date is before listed emperors.

Example

YearName(DateField; 0) Returns 平成 14 when DateField contains 7/17/2002.
Design functions return information about the structure of open database files. For example, you could determine the names of all the layouts or fields in an open database file.

**Note** FileMaker Pro limits the information returned by a design function, according to the privilege set in effect when the function evaluates a database file. See FileMaker Pro help for more information about granting access to database files.

Design function parameters can be any of the following:

- filenames such as "Customer" or literal text such as "Jack"
- field such as `layoutName`
- other functions such as `Left(text;number)`

**Important** Literal text parameters such as filenames and layout names must be enclosed in quotation marks. Use quotation marks around field names to indicate the literal string is the parameter (omit quotation marks to indicate the value stored in the field is the parameter). You can use spaces before or after the parentheses that enclose parameters, but spaces are not necessary. Use a semicolon between parameters when a function requires more than one parameter.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DatabaseNames</code>, page 43</td>
<td>A list of the names of all database files open on the computer (including files opened as a <em>client</em>), separated by carriage returns.</td>
</tr>
<tr>
<td><code>FieldBounds</code>, page 43</td>
<td>The location of each side of the specified field and its rotation in degrees.</td>
</tr>
<tr>
<td><code>FieldComment</code>, page 44</td>
<td>The specified field’s comment.</td>
</tr>
<tr>
<td><code>FieldIDs</code>, page 45</td>
<td>A list of all field IDs in the specified database file and layout, separated by carriage returns.</td>
</tr>
<tr>
<td><code>FieldNames</code>, page 45</td>
<td>A list of the names of all fields on the specified layout, separated by carriage returns.</td>
</tr>
<tr>
<td><code>FieldRepetitions</code>, page 46</td>
<td>The number of repetitions of the specified field as it is formatted on the specified layout (which could be different from the number of repetitions specified when the field was defined), and the orientation of the field repetitions (horizontal or vertical) on the layout.</td>
</tr>
<tr>
<td><code>FieldStyle</code>, page 47</td>
<td>The formatting applied to the specified field on the specified layout.</td>
</tr>
<tr>
<td><code>FieldType</code>, page 48</td>
<td>Information about the specified field.</td>
</tr>
<tr>
<td><code>GetNextSerialValue</code>, page 49</td>
<td>The next <em>serial number</em> for the specified field in the specified database file.</td>
</tr>
<tr>
<td><code>LayoutIDs</code>, page 49</td>
<td>A list of all layout IDs in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td>This function</td>
<td>Returns</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LayoutNames, page 50</td>
<td>A list of the names of all layouts in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td>LayoutObjectNames, page 50</td>
<td>A list of the names of all named layout objects, separated by carriage returns.</td>
</tr>
<tr>
<td>RelationInfo, page 51</td>
<td>A list of four values for each relationship directly related to the specified table.</td>
</tr>
<tr>
<td>ScriptIDs, page 52</td>
<td>A list of all script IDs in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td>ScriptNames, page 53</td>
<td>A list of the names of all scripts in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td>TableIDs, page 53</td>
<td>A list of all table IDs in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td>TableName, page 54</td>
<td>A list of the names of all defined tables in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td>ValueListIDs, page 54</td>
<td>A list of all value list IDs in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td>ValueListItems, page 55</td>
<td>A list of the values in the specified value list, separated by carriage returns.</td>
</tr>
<tr>
<td>ValueListNames, page 55</td>
<td>A list of the names of all value lists in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td>WindowNames, page 56</td>
<td>A list of the names of open windows in the specified database file.</td>
</tr>
</tbody>
</table>
DatabaseNames

**Format**

DatabaseNames

**Parameter**

None

**Data type returned**

text

**Description**

Returns a list of the names of all database files open on the computer, separated by carriage returns. The names do not include file extensions.

**Note** If your database is hosted on another computer, DatabaseNames returns a list of the names of local client and remote database files open only on the client computer.

**Example**

To determine whether “Customers” is one of the files currently open, use the DatabaseNames function with the Position function in the formula: Position(DatabaseNames;“Customers”;1;1). If the formula returns a value greater than 0, then Customers is open.

FieldBounds

**Format**

FieldBounds(fileName;layoutName;fieldName)

**Parameters**

fileName - the name of an open database file (local or remote).

layoutName - the name of a layout in the specified database file.

fieldName - the name of a field on the specified layout.

**Important** See “Design functions” on page 41 for information about literal text parameters.

**Data type returned**

text
Description

Returns in a non-repeating text field the location in pixels of each side of fieldName and its rotation in degrees on layoutName in the fileName file. The location is measured from the top left corner of the layout (regardless of printer margins) and is specified in this order: position of left field boundary, position of top field boundary, position of right field boundary, position of bottom field boundary, degree of rotation (measured in a counterclockwise direction; 0 degrees for unrotated).

Note Your layout begins where your margins end. Because field boundaries are measured from the left side and top of the layout, boundaries returned by FieldBounds never change unless you move or re-size a field.

Example

FieldBounds("Customers";"Layout #1";"Field") returns 36 48 295 65 0 in the example below. Notice that all parameters are enclosed in quotation marks.

FieldComment

Format

FieldComment(fileName;fieldName)

Parameters

fileName - the name of an open database file (local or remote).

fieldName - the name of a field in the specified database file.

Important See “Design functions” on page 41 for information about literal text parameters.

Data type returned
text
**Description**

Returns the specified field's comment. The field name must be in the form `tablename::fieldname` to specify a field that exists in a table different from the current table.

**Examples**

FieldComment("Customers"); "Phone Number") returns "**Customer's home telephone number**" if it was entered as a comment for the Phone Number field.

FieldComment("Customers"); "Accounts::Current Balance") returns "**Customer's current balance**" if it was entered as a comment for the Current Balance field in the Accounts table.

**FieldIDs**

**Format**

FieldIDs(fileName;layoutName)

**Parameters**

fileName - the name of an open database file (local or remote).

layoutName - the name of a layout or table in the specified database file.

**Important** See “Design functions” on page 41 for information about literal text parameters.

**Data type returned**

text

**Description**

Returns a list of all field IDs in fileName and layoutName, separated by carriage returns. Related fields are returned as TableID::RelatedFieldID.

For example, 12::4, where 12 is the ID of the table and 4 is the ID of the related field.

If layoutName is empty, then the field IDs of the default table will be returned.

**Examples**

FieldIDs("Customers";"") returns IDs of all unique fields in the default table of Customers.

FieldIDs("Customers";"Layout#5") returns IDs of all unique fields, including related fields, on Layout#5 in Customers.

**FieldNames**

**Format**

FieldNames(fileName;layoutName)
Parameters

fileName - the name of an open database file (local or remote).
layoutName - the name of a layout or table in the specified database file.

Important See “Design functions” on page 41 for information about literal text parameters.

Data type returned
text

Description

Returns a list of the names of all fields on layoutName, in fileName file, separated by carriage returns. Related fields are displayed in tablename::fieldname format.

If layoutName isn’t specified, then the field names of the first table created (the “default table”) will be returned.

Note If FieldNames returns a question mark (?) or the name of only one field, go to the Specify Calculation dialog box and make sure the Calculation result is text. Also, you can increase the size of the field on the layout to show more field names.

Examples

FieldNames(“Customers”;“”) returns a list of all the fields in the default table of the Customers database file.

FieldNames(“Customers”;“Data Entry”) returns a list of all the fields, including related fields, in the Customers database file that appear on the Data Entry layout.

FieldRepetitions

Format

FieldRepetitions(fileName;layoutName;fieldName)

Parameters

fileName - the name of an open database file (local or remote).
layoutName - the name of a layout in the specified database file.
fieldName - the name of a field on the specified layout.

Important See “Design functions” on page 41 for information about literal text parameters.

Data type returned
text
Description

Returns the number of repetitions of the repeating field fieldName as it is currently formatted on layoutName (which could be different from the number of repetitions when the field was defined), and the orientation of the field repetitions (horizontal or vertical) on the layout. If fieldName isn’t a repeating field, it returns 1 vertical.

Example

FieldRepetitions(“Customers”;“Data Entry”;“Business Phone”) returns 3 vertical if the Business Phone field is defined as a repeating field with five repetitions but is formatted to only show three repetitions in a vertical orientation on the Data Entry layout.

FieldStyle

Format

FieldStyle(fileName;layoutName;fieldName)

Parameters

fileName - the name of an open database file (local or remote).
layoutName - the name of a layout in the specified database file.
fieldName - the name of a field on the specified layout.

Important See “Design functions” on page 41 for information about literal text parameters.

Data type returned
text

Description

Returns the field formatting applied to fieldName on layoutName in the fileName file. If the field has a value list associated with it, the FieldStyle function also returns the name of the value list.

• A standard field returns Standard.
• A standard field with a vertical scroll bar returns Scrolling.
• A drop-down list returns Popuplist.
• A pop-up menu returns Popupmenu.
• A Checkbox returns Checkbox.
• A Radio button returns RadioButton.
Example
On the Data Entry layout in the Customers database file, FieldStyle("Customers";"Data Entry";"Current Customer") returns RadioButton Yes/No List when the Current Customer field is formatted as a radio button and is associated with the value list named Yes/No List.

FieldType

Format
FieldType(fileName;fieldName)

Parameters
fileName - the name of an open database file (local or remote).
fieldName - the name of a field in the specified database file.

Important See “Design functions” on page 41 for information about literal text parameters.

Data type returned
text

Description
Returns information about fieldName. Field names must be in the form tablename::fieldname to specify a field that exists in a table different from the current table. The result has four values separated by spaces:

• The first value is either Standard, StoredCalc, Summary, UnstoredCalc, or Global.
• The second value is the field type: text, number, date, time, timestamp, or container.
• The third value is Indexed or Unindexed.
• The fourth value is the maximum number of repetitions defined for the field (if the field isn’t defined as a repeating field, this value is 1).

Examples
FieldType("Customers";"Phone Number") returns Standard Text Unindexed 3, when, in the Customers database file, the Phone Number field is defined as a text field that repeats a maximum of 3 times and the storage options are left unchanged. (Most fields are indexed when a find is performed in that field.)
FieldType("Customers";"Current Balance") returns StoredCalc Number Indexed 1, when, in the Customers database file, the Current Balance field is defined as a stored, numeric calculation field that is indexed.
FieldType("Customers";"Today’s Date") returns Global Date Unindexed 1, when, in the Customers database file, the Today’s Date field is defined as a global field of type date. Global fields are never indexed.
GetNextSerialValue

**Format**

GetNextSerialValue(fileName;fieldName)

**Parameters**

- **fileName** - the name of an open database file (local or remote).
- **fieldName** - the name of the field whose next serial number you want to determine.

**Important** See “Design functions” on page 41 for information about literal text parameters.

**Data type returned**

- text

**Description**

Returns the next serial number of fieldName in fileName. Field names must be fully qualified in the form tablename::fieldname to specify a field that exists in a table different from the current table.

**Example**

GetNextSerialValue(“Customers”;”CustID”) returns the next serial number for the CustID field.

LayoutIDs

**Format**

LayoutIDs(fileName)

**Parameter**

- **fileName** - the name of an open database file (local or remote).

**Important** See “Design functions” on page 41 for information about literal text parameters.

**Data type returned**

- text

**Description**

Returns a list of all layout IDs in fileName, separated by carriage returns.

**Example**

LayoutIDs(“Customers”) returns a list of all the layout IDs in the Customers database file.
LayoutNames

Format
LayoutNames(fileName)

Parameter
fileName - the name of an open database file (local or remote).

Important See “Design functions” on page 41 for information about literal text parameters.

Data type returned
text

Description
Returns a list of the names of all layouts in fileName, separated by carriage returns.

Example
LayoutNames(“Customers”) returns a list of all the layouts in the Customers database file.

LayoutObjectNames

Format
LayoutObjectNames(fileName;layoutName)

Parameter
fileName - the name of an open database file (local or remote).

layoutName - the name of a layout in the specified database file.

Important See FileMaker Pro Help for information about literal text parameters.

Data type returned
text

Description
Returns a list of the names of all named objects on layoutName in fileName, separated by carriage returns. Layout objects without object names are not returned.

If layoutName isn’t specified, then no object names are returned.

Named tab controls, grouped objects, and portal objects that contain other named objects are followed by a list of those named objects enclosed in angle brackets (<>). The angle brackets are shown even if there are no named objects contained within the named tab controls, grouped objects, or portal objects.
Example
LayoutObjectNames ("Customers";"Data Entry") returns a list of named objects in the Customers database file that appear on the Data Entry layout.

RelationInfo

Format
RelationInfo(fileName;tableName)

Parameters
fileName - the name of an open database file (local or remote).
tableName - the name of a table in the specified database file.

Important See FileMaker Pro Help for information about literal text parameters.

Data type returned
text

Description
Returns a list of four values for each relationship directly related to tableName. Values in a list are separated by carriage returns, and lists are separated by two carriage returns. For each additional relationship connected to tableName, an additional list of four values is output.

The four values are:

• Source: the name of the database file that holds tableName
• Table: the name of the table connected to tableName
• Options: the options that were set in the right side of the Edit Relationship dialog box when the relationship was defined. This line is blank if the following options are not set; otherwise these options are separated by spaces.
  • Delete, if Delete related records in this table when a record is deleted in the other table is selected in the right side of the Edit Relationship dialog box
  • Create, if Allow creation of records in this table via this relationship is selected in the right side of the Edit Relationship dialog box
  • Sorted, if Sort records is selected in the right side of the Edit Relationship dialog box
• Relationships: a list of the defined relationships, one per line. Field names are fully qualified, for example, TableName::Field Name.
Example

A database file called Human Resources has three tables: Company, Employees, and Addresses. Company::Company ID is connected to Employees::Company ID, Employees::Employee ID is connected to Addresses::Employee ID and Employees::DateOfHire is connected to Addresses::DateMovedIn.

The relationships have the following criteria:

- You can create records in all tables.
- You cannot delete records in all tables.
- A sort was specified for the Addresses table for the Employees<-->Addresses relationship.

RelationInfo(“Human Resources”;“Employees”) returns:

Source: Human Resources
Table: Company
Options: Create
Company::Company ID = Employees::Company ID

Source: Human Resources
Table: Addresses
Options: Create Sorted
Addresses::Employee ID = Employees::Employee ID
Addresses::DateMovedIn >= Employees::DateOfHire

ScriptIDs

Format
ScriptIDs(fileName)

Parameter
c

fileName - the name of an open database file (local or remote).

Important See “Design functions” on page 41 for information about literal text parameters.

Data type returned
text

Description
Returns a list of all script IDs in fileName, separated by carriage returns.
**Example**
ScriptIDs(“Customers”) returns a list of all the script IDs in the Customers database file.

**ScriptNames**

**Format**
ScriptNames(fileName)

**Parameter**
fileName - the name of an open database file (local or remote).

**Important** See “Design functions” on page 41 for information about literal text parameters.

**Data type returned**
text

**Description**
Returns a list of the names of all scripts in fileName, separated by carriage returns.

**Example**
ScriptNames(“Customers”) returns a list of all the scripts in the Customers database file.

**TableIDs**

**Format**
TableIDs(fileName)

**Parameter**
fileName - the name of an open database file (local or remote).

**Important** See “Design functions” on page 41 for information about literal text parameters.

**Data type returned**
text

**Description**
Returns a list of all table IDs in fileName, separated by carriage returns.

**Note** Each table ID is unique. Also, the ID is independent of when you create each table: the first table could have the smallest, middle, or largest value.
**Example**
TableIDs(“University Database”) returns 1065089 1065090 for the University Database database file if two tables have been defined for the file.

**TableNames**

**Format**
TableNames(fileName)

**Parameter**
fileName - the name of an open database file (local or remote).

**Important** See “Design functions” on page 41 for information about literal text parameters.

**Data type returned**
text

**Description**
Returns a list of all table occurrences in the relationships graph for fileName, separated by carriage returns.

**Example**
TableNames(“University Database”) returns table occurrences Teachers Coaches for the University Database database file if a Teachers table and a Coaches table have been defined for the file.

**ValueListIDs**

**Format**
ValueListIDs(fileName)

**Parameter**
fileName - the name of an open database file (local or remote).

**Important** See “Design functions” on page 41 for information about literal text parameters.
**Data type returned**
text

**Description**
Returns a list of all value list IDs in fileName, separated by carriage returns.

**Example**
ValueListIDs(“Customers”) returns a list of all the value list IDs in the Customers database file.

**ValueListItems**

**Format**
ValueListItems(fileName;valuelist)

**Parameters**
fileName - the name of an open database file (local or remote).
valuelist - the name of a value list in the specified database file.

**Data type returned**
text

**Description**
Returns a list of the values in valuelist, separated by carriage returns.

**Example**
ValueListItems(“Customers”;“Code”) returns a list of all the items in the Code value list in the Customers database file.

**ValueListNames**

**Format**
ValueListNames(fileName)

**Parameter**
fileName - the name of an open database file (local or remote).

**Data type returned**
text
**Description**

Returns a list of the names of all value lists in fileName, separated by carriage returns.

**Example**

ValueListNames(“Customers”) returns a list of all the value list names in the Customers database file.

---

**WindowNames**

**Format**

WindowNames{(fileName)}

**Parameter**

{fileName} - the name of an open database file (local or remote).

Parameters in curly braces {} are optional.

**Data type returned**

text

**Description**

Returns a list of the names of windows that are currently open. Use the optional fileName parameter to only return windows that are based on the specified file. The window could be visible, hidden, or minimized. The order of the names in the list matches the current stacking order of the windows. The visible windows are listed first, then the minimized windows, then the hidden windows. If there are no databases or windows open, an empty string is returned.

**Note**  Even if you close a file, it may remain open as a hidden file if the window of any other file is displaying data from that file. (For example, another window may be displaying related data from the file you attempted to close.) FileMaker Pro will close the file when you close all the dependent windows.

**Examples**

WindowNames returns Customers and Invoices separated by a carriage return when those windows are currently open.

WindowNames(“contacts”) returns a list of windows that are based on the contacts database file.
Use external functions to access FileMaker Pro plug-ins. Plug-ins add features to FileMaker Pro. For more information, see FileMaker Pro help.

External functions are only available if FileMaker Pro plug-ins are installed and enabled on your computer. If no FileMaker Pro plug-ins are installed, you see only the generic external function definition in the Specify Calculation dialog box:

External (nameOfFunction; parameter)

**Plug-ins written for version 7.0 and later**

Each plug-in defines its own functions and parameters. See the documentation that came with the plug-in for each function’s usage.

**Plug-ins written for version 6.0 and earlier**

These plug-ins are still supported and continue to use the External function to access the plug-in’s functions. The first parameter is the name of the plug-in function to execute and the second is a parameter that is passed to that function. See the documentation that came with the plug-in for each function’s usage.

<table>
<thead>
<tr>
<th>This function</th>
<th>Does this</th>
</tr>
</thead>
<tbody>
<tr>
<td>External, page 58</td>
<td>Enables access to FileMaker Pro plug-ins written for versions of FileMaker Pro prior to 7.0.</td>
</tr>
</tbody>
</table>

For more information, see FileMaker Pro help.
External

Format

External(nameOfFunction;parameter)

Parameters

nameOfFunction - the name of the external function

parameter - the parameter(s) required by the external function. A parameter is required, even if it’s only 0.

Data type returned

Depends on the external function

Description

The External function accesses plug-ins created for versions of FileMaker Pro prior to 7.0 and uses the syntax External(“function name”, parameter), where function name is in quotes and is the name of an external function.

Plug-ins created for FileMaker Pro version 7.0 and later do not use the External(“function name”, parameter) syntax. For more information, see FileMaker Pro Help and the FileMaker Pro Advanced Development Guide included with FileMaker Pro Advanced.
Chapter 6
Financial functions

Financial functions calculate financial information, such as net present value and payments. For example, you can calculate the monthly payments required to buy a car at a certain loan rate using the PMT function.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV, page 60</td>
<td>The future value of an initial investment, based on a constant interest rate and payment amount for the number of periods in months.</td>
</tr>
<tr>
<td>NPV, page 60</td>
<td>The net present value of a series of unequal payments made at regular intervals, assuming a fixed rate per interval.</td>
</tr>
<tr>
<td>PMT, page 61</td>
<td>The payment required to meet the requirements of the term, interest rate, and principal.</td>
</tr>
<tr>
<td>PV, page 62</td>
<td>The present value of a series of equal payments made at regular intervals (periods), assuming a fixed interest rate per interval.</td>
</tr>
</tbody>
</table>
FV

Format
FV(payment; interestRate; periods)

Parameters
payment - payment to be made per period
interestRate - interest rate per period
periods - number of periods

Data type returned
number

Description
Returns the future value of an initial investment, based on a constant interestRate and payment amount for the number of periods in months. For example, you can calculate how much you’ll earn on an investment in which you pay $50 a month for 60 months at a 6 percent annual interest rate.

Notes
• When interestRate is 0, this function returns the result of payment * periods.
• The FV function doesn’t account for the present value of your investment, and it assumes that payment is made at the end of each period.

\[
FV = payment \times \frac{(1 + interestRate)^{periods} - 1}{interestRate}
\]

Examples
FV(50; .11/12; 5 * 12) returns 3975.90398429....
FV(2000; .12; 30) + 5000 \times (.12 + 1) ^ 30 returns 632464.97928640....
FV(500; .11/5; 60) returns 61141.65130790....

To set the decimal precision of the returned value, enclose the current formulas with the Round function. For example, Round(Current Formula; 2).

NPV

Format
NPV(payment; interestRate)

Parameters
payment - a repeating field containing unequal payment amounts, or an expression that returns a reference to one.
interestRate - interest rate.

Data type returned
number

Description
Returns the net present value of a series of unequal payments made at regular intervals, assuming a fixed interestRate per interval. For example, suppose someone borrows money from you and pays you back in unequal amounts over a period of several years. You can calculate the result using the NPV function.

\[
NPV = \sum_{i=1}^{n} \frac{-loan \text{ amount}}{(1 + \text{interestRate})^i} + \sum_{i=1}^{n} \frac{-\text{first payment}}{(1 + \text{interestRate})^2} + \sum_{i=1}^{n} \frac{-\text{second payment}}{(1 + \text{interestRate})^3} + \ldots + \sum_{i=1}^{n} \frac{-n^{\text{th}} \text{ payment}}{(1 + \text{interestRate})^{n+1}}
\]

Examples
NPV(Loan; .05) returns 156.91277445..., when the repeating field, Loan, contains -2000 (the initial payment), 600, 300, 500, 700, and 400. The result (156.91277445...) is the actual profit in today’s dollars that will be realized from this transaction.

NPV(Amounts; .10) returns 16758.35604870..., when the repeating field, Amounts, contains -5000 (the initial investment), 10,000, 0, 10,000, and 10,000.

If you want each return value to return 2 decimal places, surround the current formulas with the correct Round function: Round(Current Formula;2).

PMT

Format
PMT(principal;interestRate;term)

Parameters
principal - principal amount.
interestRate - interest rate. If the interest rate is annual, divide the rate by 12.
term - length of time, expressed in number of months.

Data type returned
number

Description
Returns the payment required to meet the requirements of the term, interestRate, and principal you supply.

\[
PMT = \frac{payment \cdot (1 - (1 + \text{interestRate})^{-\text{periods}})}{\text{interestRate}}
\]
Examples
In the following example, the \texttt{PMT} function calculates payments for purchasing a sports car costing $21,000, at an annual rate of 6.9\% over 48 monthly payments.

\texttt{PMT(21000;.069/12;48)} returns the payment amount $501.90.

\texttt{PMT(Cost;.13;Years)} returns a payment amount, based on the purchase value stored in \texttt{Cost}, at a 13\% rate, over the duration stored in \texttt{Years}.

\texttt{“Your payment will be “ & PMT(150000;.13/12;Months) & “.”} returns \texttt{Your payment will be}, followed by the payment amount, based on a total cost of $150,000, at a 13\% annual percentage rate, over the duration stored in \texttt{Months}.

\textbf{PV}

\textbf{Format}

\texttt{PV(payment;interestRate;periods)}

\textbf{Parameters}

- \texttt{payment} - payment amount to be made per period. Type a negative number for money you pay and a positive number for money you receive.
- \texttt{interestRate} - interest rate per period.
- \texttt{periods} - number of periods (intervals between payments).

\textbf{Data type returned}

number

\textbf{Description}

Returns the present value of a series of equal payments made at regular intervals (periods), assuming a fixed interest rate per interval.

\[ PV = \text{payment} \times \frac{1-(1+\text{interestRate})^{-\text{periods}}}{\text{interestRate}} \]

\textbf{Note} When \texttt{interestRate} is 0, this function returns the result of \texttt{payment} \times \texttt{periods}.

\textbf{Example}

Your cousin borrowed $2,000 from you, offering to pay you back $500 a year for five years, for a total of $2,500 at the end of five years. If inflation was 5\% annually, with the following entry you could find out what those payments are worth with the \texttt{PV} function.

\texttt{PV(500;.05;5)} returns 2164.73833531...

If you want the return value to return two decimal places, enclose the formula with the correct \texttt{Round} function: \texttt{Round(\text{Current Formula};2)}.
Chapter 7
Get functions

Use Get functions in scripts for error checking and prevention, or to capture information about the status of a database file or elements in it, or an action being performed.

Many Get functions return information that changes on a regular basis. For example, when the Get(CurrentTime) function is placed in a stored calculation field, the time will only update when a new record is created. If the calculation has other fields in it, but the calculation result still returns the current time, then the stored calculation result will only update when those other fields have been modified in the current record. If either of these calculations are unstored, the time will update as needed. For performance reasons, making a calculation field unstored is not always the best idea. Get functions are best used in a script where the status information from a Get function is up to date at the moment that the calculation is run.

To access the list of Get functions, in the Specify Calculation dialog box, choose View all functions by type or View Get functions. When you View all functions by name, you see only Get(flag).

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get(AccountName), page 68</td>
<td>The authenticated account name being used for the active database file.</td>
</tr>
<tr>
<td>Get(ActiveFieldContents), page 68</td>
<td>The contents of the field containing the cursor.</td>
</tr>
<tr>
<td>Get(ActiveFieldName), page 69</td>
<td>The name of the field currently containing the cursor.</td>
</tr>
<tr>
<td>Get(ActiveFieldTableNames), page 69</td>
<td>The name of the table that contains the active field (the field that currently contains the cursor).</td>
</tr>
<tr>
<td>Get(ActiveLayoutObjectName), page 70</td>
<td>The name of the layout object in the calculation's active window (the window that currently contains the cursor).</td>
</tr>
<tr>
<td>Get(ActiveModifierKeys), page 71</td>
<td>A number representing the keyboard modifier keys (for example, Shift) that are being pressed.</td>
</tr>
<tr>
<td>Get(ActiveRepetitionNumber), page 71</td>
<td>A number representing the current (active) repetition of a repeating field.</td>
</tr>
<tr>
<td>Get(ActiveSelectionSize), page 72</td>
<td>A number representing how many characters are selected.</td>
</tr>
<tr>
<td>Get(ActiveSelectionStart), page 72</td>
<td>A number representing the starting character of the selected text.</td>
</tr>
<tr>
<td>Get(AllowAbortState), page 73</td>
<td>A Boolean value representing the current state of Allow user abort script step.</td>
</tr>
<tr>
<td>Get(AllowToolbarState), page 73</td>
<td>A Boolean value representing whether toolbars are allowed to be visible.</td>
</tr>
<tr>
<td>Get(ApplicationLanguage), page 74</td>
<td>Text representing the current application language (for example, English).</td>
</tr>
<tr>
<td><strong>This function</strong></td>
<td><strong>Returns</strong></td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Get(ApplicationVersion), page 75</td>
<td>Text representing the FileMaker Pro application version.</td>
</tr>
<tr>
<td>Get(CalculationRepetitionNumber), page 75</td>
<td>A number representing the repetition of the calculation field that is currently being calculated.</td>
</tr>
<tr>
<td>Get(CurrentDate), page 76</td>
<td>The current date according to the system calendar.</td>
</tr>
<tr>
<td>Get(CurrentHostTimestamp), page 76</td>
<td>The host's current date and time (to the nearest second) according to the system clock.</td>
</tr>
<tr>
<td>Get(CurrentTime), page 77</td>
<td>The current time (to the nearest second) according to the system clock.</td>
</tr>
<tr>
<td>Get(CurrentTimestamp), page 78</td>
<td>The current date and time (to the nearest second) according to the system clock.</td>
</tr>
<tr>
<td>Get(CustomMenuSetName), page 78</td>
<td>The name of the active custom menu set.</td>
</tr>
<tr>
<td>Get(DesktopPath), page 79</td>
<td>The path to the desktop folder for the current user.</td>
</tr>
<tr>
<td>Get(DocumentsPath), page 80</td>
<td>The path to the documents folder for the current user.</td>
</tr>
<tr>
<td>Get(ErrorCaptureState), page 80</td>
<td>A Boolean value representing the state of Error capture script step.</td>
</tr>
<tr>
<td>Get(ExtendedPrivileges), page 81</td>
<td>A list of key words for the enabled extended privileges, separated by carriage returns.</td>
</tr>
<tr>
<td>Get(FileMakerPath), page 81</td>
<td>The path to the folder of the currently running copy of FileMaker Pro.</td>
</tr>
<tr>
<td>Get(FileName), page 82</td>
<td>The name of the currently active database file.</td>
</tr>
<tr>
<td>Get(FilePath), page 82</td>
<td>The full path indicating the location of the file.</td>
</tr>
<tr>
<td>Get(FileSize), page 83</td>
<td>The size (in bytes) of the currently active database file.</td>
</tr>
<tr>
<td>Get(FoundCount), page 84</td>
<td>A number that represents the number of records in the current found set.</td>
</tr>
<tr>
<td>Get(HighContrastColor), page 84</td>
<td>The name of the current high contrast default color scheme if Use High Contrast is selected in the Windows operating system Accessibility Options dialog box.</td>
</tr>
<tr>
<td>Get(HighContrastState), page 85</td>
<td>A Boolean value representing the state of the Use High Contrast checkbox on the Accessibility Options dialog box.</td>
</tr>
<tr>
<td>Get(HostIPAddress), page 85</td>
<td>The IP address of the host machine for the current database.</td>
</tr>
<tr>
<td>Get(HostName), page 86</td>
<td>The registered name of the computer that is hosting the database file.</td>
</tr>
<tr>
<td>Get(LastError), page 87</td>
<td>A number representing the error, if any, in the execution of the most recently executed script step.</td>
</tr>
<tr>
<td>Get(LastMessageChoice), page 93</td>
<td>A number corresponding to the button clicked in an alert message displayed by the Show Custom Dialog script step.</td>
</tr>
<tr>
<td>Get(LastODBCError), page 94</td>
<td>A string that shows the error state published by ODBC standards, based on ISO/IEF standards.</td>
</tr>
<tr>
<td>Get(LayoutAccess), page 94</td>
<td>A number corresponding to the layout access privileges assigned through the Define Accounts &amp; Privileges dialog box.</td>
</tr>
<tr>
<td>Get(LayoutCount), page 95</td>
<td>The total number of layouts in the database file.</td>
</tr>
<tr>
<td>Get(LayoutName), page 96</td>
<td>The name of the layout currently displayed.</td>
</tr>
<tr>
<td>Get(LayoutNumber), page 96</td>
<td>The number of the layout currently displayed, according to the list in the Set Layout Order dialog box.</td>
</tr>
<tr>
<td>This function</td>
<td>Returns</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Get(LayoutTableName), page 97</td>
<td>The name of the table that the layout is displaying records from.</td>
</tr>
<tr>
<td>Get(LayoutViewState), page 97</td>
<td>Information about how the database file is being viewed.</td>
</tr>
<tr>
<td>Get(MultiUserState), page 98</td>
<td>A number representing the current multi-user state of the database file.</td>
</tr>
<tr>
<td>Get(NetworkProtocol), page 99</td>
<td>The name of the network protocol that FileMaker Pro is using on this machine.</td>
</tr>
<tr>
<td>Get(PageNumber), page 99</td>
<td>A number representing the current page being printed or previewed.</td>
</tr>
<tr>
<td>Get(PortalRowNumber), page 100</td>
<td>The number of the currently selected portal row.</td>
</tr>
<tr>
<td>Get(PreferencesPath), page 100</td>
<td>The path to the preferences folder for the current user.</td>
</tr>
<tr>
<td>Get(PrivilegeSetName), page 102</td>
<td>The name of the privilege set being used by the current user.</td>
</tr>
<tr>
<td>Get(RecordAccess), page 102</td>
<td>A number indicating the access privileges of the current record.</td>
</tr>
<tr>
<td>Get(RecordID), page 103</td>
<td>The unique ID number of the current record.</td>
</tr>
<tr>
<td>Get(RecordModificationCount), page 104</td>
<td>The total number of times changes to the current record have been committed.</td>
</tr>
<tr>
<td>Get(RecordNumber), page 105</td>
<td>The number of the current record in the current found set.</td>
</tr>
<tr>
<td>Get(RecordOpenCount), page 105</td>
<td>The total number of open records in the current found set that haven’t yet been saved.</td>
</tr>
<tr>
<td>Get(RecordOpenState), page 106</td>
<td>A number representing the state of the current record.</td>
</tr>
<tr>
<td>Get(RequestCount), page 107</td>
<td>The total number of find requests currently defined for the current table.</td>
</tr>
<tr>
<td>Get(RequestOmitState), page 107</td>
<td>A Boolean value representing the state of the Omit checkbox in Find mode.</td>
</tr>
<tr>
<td>Get(ScreenDepth), page 108</td>
<td>The number of bits needed to represent the color or shade of gray of a pixel on the main screen.</td>
</tr>
<tr>
<td>Get(ScreenHeight), page 108</td>
<td>The number of pixels displayed vertically on the screen in which the window of the current file is open.</td>
</tr>
<tr>
<td>Get(ScreenWidth), page 109</td>
<td>The number of pixels displayed horizontally on the screen in which the window of the current file is open.</td>
</tr>
<tr>
<td>Get(ScriptName), page 109</td>
<td>The name of the script currently running (or paused).</td>
</tr>
<tr>
<td>Get(ScriptParameter), page 110</td>
<td>The script parameter passed into the current script.</td>
</tr>
<tr>
<td>Get(ScriptResult), page 111</td>
<td>The script result from a performed subscript.</td>
</tr>
<tr>
<td>Get(SortState), page 112</td>
<td>A Boolean value representing the current sort state.</td>
</tr>
<tr>
<td>Get(StatusAreaState), page 112</td>
<td>A number representing whether the status area is hidden, visible, visible and locked, or hidden and locked.</td>
</tr>
<tr>
<td>Get(SystemDrive), page 113</td>
<td>The drive letter (Windows) or the volume name (Mac OS) where the currently running operating system is located.</td>
</tr>
<tr>
<td>Get(SystemIPAddress), page 114</td>
<td>The IP addresses of all the machines connected to a NIC (Network Interface Controller) card.</td>
</tr>
<tr>
<td>Get(SystemLanguage), page 114</td>
<td>The language currently set on the current system.</td>
</tr>
<tr>
<td>Get(SystemNICAddress), page 115</td>
<td>The hardware addresses of all the Network Interface Controller cards connected to the machine.</td>
</tr>
<tr>
<td>This function</td>
<td>Returns</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Get(SystemPlatform), page 115</td>
<td>A number indicating the current platform.</td>
</tr>
<tr>
<td>Get(SystemVersion), page 116</td>
<td>The version of the operating system of the machine on which the function is executed.</td>
</tr>
<tr>
<td>Get(TextRulerVisible), page 116</td>
<td>A Boolean value representing whether or not the text ruler is visible.</td>
</tr>
<tr>
<td>Get(TotalRecordCount), page 117</td>
<td>The total number of records in the current table.</td>
</tr>
<tr>
<td>Get(UserCount), page 117</td>
<td>The number of users who are currently accessing the file.</td>
</tr>
<tr>
<td>Get(UserName), page 118</td>
<td>The name of the FileMaker Pro user, as specified in the General tab of the Preferences dialog box.</td>
</tr>
<tr>
<td>Get(UseSystemFormatsState), page 119</td>
<td>A Boolean value representing the state of the Use System Formats menu command.</td>
</tr>
<tr>
<td>Get(WindowContentHeight), page 119</td>
<td>A number representing the height, in pixels, of the content area.</td>
</tr>
<tr>
<td>Get(WindowContentWidth), page 120</td>
<td>A number representing the width, in pixels, of the content area.</td>
</tr>
<tr>
<td>Get(WindowDesktopHeight), page 120</td>
<td>A number representing the height, in pixels, of the desktop space.</td>
</tr>
<tr>
<td>Get(WindowDesktopWidth), page 121</td>
<td>A number representing the width, in pixels, of the desktop space.</td>
</tr>
<tr>
<td>Get(WindowHeight), page 122</td>
<td>A number representing the height, in pixels, of the current window of the file in which the calculation is defined.</td>
</tr>
<tr>
<td>Get(WindowLeft), page 122</td>
<td>A number representing the horizontal distance, in pixels, of the outer edge of the current window relative to the left-most edge of the screen.</td>
</tr>
<tr>
<td>Get(WindowMode), page 123</td>
<td>A number representing whether FileMaker Pro is in Browse mode, Find mode, Preview mode, or printing when the function is evaluated.</td>
</tr>
<tr>
<td>Get(WindowName), page 123</td>
<td>The name of the current window of the file in which the calculation is defined.</td>
</tr>
<tr>
<td>Get(WindowTop), page 124</td>
<td>A number representing the vertical distance, in pixels, of the outer edge of the current window relative to the bottom edge of the menu bar or toolbar.</td>
</tr>
<tr>
<td>Get(WindowVisible), page 125</td>
<td>A Boolean value representing whether or not the current window is visible.</td>
</tr>
<tr>
<td>Get(WindowWidth), page 125</td>
<td>A number representing the width, in pixels, of the current window of the file in which the calculation is defined.</td>
</tr>
<tr>
<td>Get(WindowZoomLevel), page 126</td>
<td>The zoom level of the current window.</td>
</tr>
</tbody>
</table>
Get functions example

This script uses the function *Get(CurrentDate)* to check each record in the found set to see if an account is past due. If an account is past due, the script shows a message and prompts the user to click a button labeled Ignore, Send letter, or Send mail (set up through the Show Custom Dialog script step). The script captures the user’s response using *Get(LastMessageChoice)*. Then, based on the user’s response, the script performs an action: it cancels the rest of the script, prints a "payment is late" letter, or sends email to the associated account.

Enter Browse Mode []
Go to Layout ["LayoutManager"]
Go to Record/Request/Page [First]
Loop
  If [DatabaseName::Date < Get(CurrentDate) - 30]
    Show Custom Dialog ["30 or more days late"]
    If [Get(LastMessageChoice) = 1]
      Halt Script
    Else If [Get(LastMessageChoice) = 2]
      Go to Layout ["Late Notice"]
      Print []
    Else
      Send Mail [To: DatabaseName::Client; Subject: "Late Notice"; Message: "Your account is past due."]
    End If
  End If
End Loop
Go to Layout [original layout]
Get(AccountName)

**Format**

Get(AccountName)

**Parameter**

None

**Data type returned**

text

**Description**

For FileMaker **authentication**, `Get(AccountName)` returns the name of the authenticated account being used by the current user of the database file. If a user is using the default Admin account, `Get(AccountName)` returns `Admin`. If a user is using the FileMaker Pro guest account then `[Guest]` will be returned.

For external server authentication, `Get(AccountName)` returns the name of the authenticated account being used by the current user of the database file, not the group the user belongs to (the group name appears in the Account list when you define accounts and privileges in FileMaker Pro). If an individual belongs to more than one group (account), the first group name listed when you View By Authentication Order while defining accounts and privileges determines access for the user.

**Notes**

- If you specify the **context** for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
- See FileMaker Pro help for information about running **scripts** in client/server and peer-to-peer environments.

**Example**

Returns `Marketing` when Marketing is the name of the account that was used to log in to the database file.

Get(ActiveFieldContents)

**Format**

Get(ActiveFieldContents)

**Parameter**

None
Data type returned
text, number, date, time, timestamp, container

Description
Returns the contents of the field containing the cursor. When the cursor is in a repeating field, returns the contents of the active repetition. The result type of the active field depends upon the data type of the active field and the result type assigned to the Get(ActiveFieldContents) calculation function.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns SomeShop when the cursor is in the Name field, and that field contains the data “SomeShop.”

This type of calculation is most useful if used in a script when you want to examine data in different fields as the script proceeds.

Get(ActiveFieldName)

Format
Get(ActiveFieldName)

Parameter
None

Data type returned
text

Description
Returns the name of the field currently containing the cursor.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns Country, when the cursor is in the Country field.

Get(ActiveFieldTableName)

Format
Get(ActiveFieldTableName)
Parameter
None

Data type returned
text

Description
Returns the name of the table that contains the active field (the field that currently contains the cursor). If there is no active field, an empty string is returned.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
There are two fields, Teachers::Name and Coaches::Name, on the current layout. Creating a script that returns the result of Get(ActiveFieldTableName) to a third field will return Teachers when the script is performed after clicking in the Teachers::Name field, or will return Coaches after clicking in the Coaches::Name field.

Get(ActiveLayoutObjectName)

Format
Get(ActiveLayoutObjectName)

Parameter
None

Data type returned
text

Description
Returns the object name of the active layout object in the calculation's current window. Otherwise returns an empty string.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
There is a named button on the current layout called cancelButton. When the cursor is on the button, Get(ActiveLayoutObjectName) returns cancelButton.
Get(ActiveModifierKeys)

Format
Get(ActiveModifierKeys)

Parameter
None

Data type returned
number

Description
Returns a number representing the keyboard modifier keys (for example, Control+Shift) that are being pressed. The number is calculated by summing numbers representing each modifier key being pressed. The values assigned to the keys are:
  • Shift = 1
  • Caps Lock = 2
  • Ctrl (Windows) and Control (Mac OS) = 4
  • Alt (Windows) and Option (Mac OS) = 8
  • ⌘ (Mac OS) = 16

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns the number 9 when Shift+Alt is pressed on a computer running Windows.
You could use this function in a script that includes a custom dialog box script step (with an OK and Cancel button) to perform some special action if the user presses the Alt (or Option) key while clicking OK.

Get(ActiveRepetitionNumber)

Format
Get(ActiveRepetitionNumber)

Parameter
None

Data type returned
number
**Description**

Returns a number representing the active repetition of a repeating field (the repetition that currently contains the cursor). The first repetition is 1. If the current field isn’t a repeating field, the function returns 1.

*Note* See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Example**

Returns 5 when the cursor is in the fifth repetition of a repeating field.

---

**Get(ActiveSelectionSize)**

**Format**

Get (ActiveSelectionSize)

**Parameter**

None

**Data type returned**

number

**Description**

Returns a number representing how many characters are selected. Returns 0 if there is no selection.

*Note* See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Example**

Returns 4 when 4 characters are selected.

---

**Get(ActiveSelectionStart)**

**Format**

Get (ActiveSelectionStart)

**Parameter**

None

**Data type returned**

number
Description
Returns a number representing the starting character of the selected text. Returns the cursor’s current position if no text is selected.
If there are multiple windows open in the current database file, a result is returned for only the foreground window.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 5 when the selection starts at character 5.

Get(AllowAbortState)

Format
Get (AllowAbortState)

Parameter
None

Data type returned
number

Description
Returns 1 if Allow user abort script step is on, otherwise returns 0.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 1 if Allow user abort script step is on.

Get(AllowToolbarState)

Format
Get (AllowToolbarState)

Parameter
None

Data type returned
number
Description

Returns a Boolean value representing whether toolbars are allowed to be visible. Returns 1 if toolbars are allowed, otherwise returns 0. The Allow Toolbars script step sets the toolbar state. For more information, see FileMaker Pro help.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example

Returns 1 if toolbars are allowed to be visible.

Get(ApplicationLanguage)

Format

Get(ApplicationLanguage)

Parameter

None

Data type returned

text

Description

Returns text representing the current application language. The text that is returned is in the English language.

FileMaker Pro supports:

• English
• French
• Italian
• German
• Swedish
• Spanish
• Dutch
• Japanese

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example

Returns English when the current application language is English.
Get(ApplicationVersion)

**Format**
Get(ApplicationVersion)

**Parameter**
None

**Data type returned**
text

**Description**
Returns text representing the FileMaker application and version.
- **Pro** *(version)* for FileMaker Pro.
- **ProAdvanced** *(version)* for FileMaker Pro Advanced.
- **Runtime** *(version)* for FileMaker Runtime.
- **Web** *(version)* for FileMaker Web Client.
- **Server** *(version)* for FileMaker Web Server.

**Note** See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Example**
Returns Pro 8.5v1 in FileMaker Pro 8.5v1.

Get(CalculationRepetitionNumber)

**Format**
Get(CalculationRepetitionNumber)

**Parameter**
None

**Data type returned**
number

**Description**
Returns a number representing the repetition of the calculation field that is currently being calculated. The first repetition is 1. If the current field isn’t a repeating field, the function returns 1.
Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 5 when FileMaker Pro is calculating the fifth repetition of a repeating field.

Get(CurrentDate)

Format
Get (CurrentDate)

Parameter
None

Data type returned
date

Description
Returns the current date according to the system calendar.

The format of the result varies based on the date format that was in use when the database file was created. In the United States, dates are generally in the format MM/DD/YYYY. You can change the date format in the Regional Settings Control Panel (Windows 2000), the Date and Time Control Panel (Windows XP), or the Date & Time System Preference (Mac OS).

If the result is displayed in a field, it is formatted according to the date format of the field in the current layout.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Important To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

Example
Returns 2/2/2004 when the system date is set to February 2, 2004.

Get(CurrentHostTimestamp)

Format
Get (CurrentHostTimestamp)

Parameter
None
Data type returned

timestamp

Description

Returns the host’s current date and time according to the system clock, to the nearest second.

The format of the value returned is determined by the database file’s settings. You can use your client system’s settings by editing the Regional Settings Control Panel (Windows 2000), the Date and Time Control Panel (Windows XP), or the Date & Time System Preference (Mac OS).

Notes

• The client machine and host machine may be in different times zones so Get(CurrentHostTimestamp) and Get(CurrentTimestamp) may return different date/time values. Also, the current date and time are characteristics of the host system, but the format of the date and time is a characteristic of the database file.

• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Important For users who are connected over a network, the Get(CurrentHostTimestamp) function can affect the performance of the database file. For example, if you use the function in an unstored calculation field, and the field is visible in a list view, each display of the field requires an additional network access. Stored calculation fields are a better use of the function. For example, if you automatically enter a timestamp for each newly created record using a stored calculation field, you minimize network access.

Example

Returns 1/1/2004 11:30:01 AM when the system clock shows January 1, 2004 11:30:01 AM on the host machine.

Get(CurrentTime)

Format

Get(CurrentTime)

Parameter

None

Data type returned

time
Description
Returns the current time according to the system clock, to the nearest second. The format of the value returned is determined by the Regional Settings Control Panel (Windows 2000), the Date and Time Control Panel (Windows XP), or the Date & Time System Preference (Mac OS).

Note In client/server and peer-to-peer environments, Get(CurrentTimestamp) evaluates the status of the client machine running the script (not the host machine). See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 11:30:00 when the system clock shows 11:30:00.

Get(CurrentTimestamp)

Format
Get (CurrentTimestamp)

Parameter
None

Data type returned
timestamp

Description
Returns the current date and time according to the system clock, to the nearest second. The format of the value returned is determined by the Regional Settings Control Panel (Windows 2000), the Date and Time Control Panel (Windows XP), or the Date & Time System Preference (Mac OS).

Note In client/server and peer-to-peer environments, Get(CurrentTimestamp) evaluates the status of the client machine running the script (not the host machine). See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 1/1/2004 11:30:00 AM when the system clock shows January 1, 2004 11:30:00.

Get(CustomMenuSetName)

Format
Get (CustomMenuSetName)
Get functions

Parameter
None

Data type returned
text

Description
Returns the name of the active custom menu set. If the active menu set isn’t a custom menu set, an empty string is returned.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns Custom Menu Set #1 when this custom menu set is active.
Returns an empty string when the [Standard FileMaker Menus] menu set is active.

Get(DesktopPath)

Format
Get(DesktopPath)

Parameter
None

Data type returned
text

Description
Returns the path to the desktop folder for the current user. In Windows, the path format is /Drive:/Documents and Settings/UserName/Desktop/. In the Mac OS, the path format is /DriveName/Users/UserName/Desktop/.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns /C:/Documents and Settings/John Smith/Desktop/ for a user named John Smith in Windows.
Returns /MacintoshHD/Users/John Smith/Desktop/ for a user named John Smith in the Mac OS.
Get(DocumentsPath)

**Format**
Get(DocumentsPath)

**Parameter**
None

**Data type returned**
text

**Description**
Returns the path to the documents folder for the current user. In Windows, the path format is /Drive:/Documents and Settings/UserName/My Documents/. In the Mac OS, the path format is /DriveName/Users/UserName/Documents/.

**Note** See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Examples**
Returns /C:/Documents and Settings/John Smith/My Documents/ for a user named John Smith in Windows.

Returns /MacintoshHD/Users/John Smith/Documents/ for a user named John Smith in the Mac OS.

Get(ErrorCaptureState)

**Format**
Get(ErrorCaptureState)

**Parameter**
None

**Data type returned**
number

**Description**
Returns 1 if the Set Error capture script step is on; otherwise returns 0.

**Note** See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Example**
Returns 1 if the Set Error capture script step is on.
Get(ExtendedPrivileges)

**Format**

Get(ExtendedPrivileges)

**Parameter**

None

**Data type returned**

text

**Description**

Returns a list of keywords, separated by carriage returns, identifying the extended privileges available to the account being used by the current user of the database file (extended privileges are additional access rights assigned to an account's privilege set). For more information, see FileMaker Pro help.

Get(ExtendedPrivileges) returns an empty list if a user doesn't have extended privileges assigned for the current database file.

**Notes**

• If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.

• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Example**

Your account uses a privilege set that includes the extended privilege of Access via Instant Web Publishing (keyword "fmiwp").

Position(Get(ExtendedPrivileges); "fmiwp"; 1; 1) returns a value greater than 0.

Get(FileMakerPath)

**Format**

Get(FileMakerPath)

**Parameter**

None

**Data type returned**

text
**Description**

Returns the path to the folder of the currently running copy of FileMaker Pro. In Windows, the path format is `/Drive:/Program Files/FileMaker/FileMaker Pro 8.5/`. In the Mac OS, the path format is `/DriveName/Applications/FileMaker Pro 8.5/`.

**Note** See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Examples**


Returns `/MacintoshHD/Applications/FileMaker Pro 8.5/` in the Mac OS.

---

**Get(FileName)**

**Format**

Get(FileName)

**Parameter**

None

**Data type returned**

text

**Description**

Returns the name of the currently active database file, without the filename extension.

**Notes**

- If the current calculation is stored and you specify its context, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
- See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Example**

Returns Contacts when Contacts is the active file.

---

**Get(FilePath)**

**Format**

Get(FilePath)
**Parameter**
None

**Data type returned**
text

**Description**
Returns the full path indicating the location of the currently active database file. In Windows, the full path is file:/drive:/folder/filename for local files. For remote files, the full path is file://volume/folder/filename. In the Mac OS, the full path is file:/volume/folder/filename for local and remote files.

**Notes**
- If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
- See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Examples**
Returns file:/driveletter:/databaseName for local files in Windows.
Returns file://volumename/myfoldername/databaseName for remote files in Windows.
Returns file:/path/databaseName for local and remote files in the Mac OS.
Returns fmnet://networkaddress/databaseName for FileMaker Pro networked files.

**Get(FileSize)**

**Format**
Get(FileSize)

**Parameter**
None

**Data type returned**
number

**Description**
Returns the size (in bytes) of the currently active database file.
Notes

• If the current calculation is stored and you specify its context, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.

• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example

Returns 15000 when the current file size is 15000 bytes.

Get(FoundCount)

Format

Get(FoundCount)

Parameter

None

Data type returned

number

Description

Returns a number that represents the number of records in the current found set. If there are multiple windows open in the current database file, each window can have its own found count value, but results are returned for only the foreground window.

Notes

• If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.

• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example

Returns 7 when there are 7 records in the current found set.

Get(HighContrastColor)

Format

Get(HighContrastColor)
Parameter
None

Data type returned
text

Description
Windows: returns the name of the current high contrast default color scheme. Returns an empty value (null) if Use High Contrast is unavailable, inactive, or if the function is used on the Mac OS.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns “High Contrast White (large)” when the Use High Contrast option in Windows 2000 is active and Black on white is selected.

Get(HighContrastState)

Format
Get(HighContrastState)

Parameter
None

Data type returned
number

Description
Windows: Returns a number representing the state of the Use High Contrast option in the Accessibility Options control panel. Returns:

- 0 if Use High Contrast is unavailable, inactive, or if the function is used on the Mac OS.
- 1 if Use High Contrast is available and active.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Get(HostIPAddress)

Format
Get(HostIPAddress)
Parameter
None

Data type returned
text

Description
Returns the IP address of the host machine for the current database. If the current database isn’t being hosted, an empty string is returned.

Notes
• If the current calculation is stored and you specify its context, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 14.156.13.121 when the current database is being hosted.

Get(HostName)

Format
Get(HostName)

Parameter
None

Data type returned
text

Description
Returns the registered name of the computer that is hosting the database file.

On the computer that is hosting the database file:
• Windows: Choose Start menu > Control Panels > System, and then click the Network Identification tab. Full computer name displays the current registered name. Click Properties to rename the computer (you must be logged on as an administrator to rename the computer).
• Mac OS: In the Sharing System Preference, Computer Name displays the current registered name.
Notes

- If the current calculation is stored and you specify its context, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
- See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example

Returns Fred Jones when Fred Jones is the registered name of the host computer in use.

Get(LastError)

Format

Get(LastError)

Parameter

None

Data type returned

number

Description

Returns a number representing the error, if any, in the execution of the most recently executed script step. Use this function to detect and control the outcome of errors.

Notes

- Mac OS: In FileMaker Pro, if an error occurs while performing an AppleScript from ScriptMaker™, the AppleScript error code will be returned.
- In FileMaker Pro, if an error occurs while performing a SQL query, a SQLSTATE error will be returned by ODBC.
- See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Tip

To create a script that responds to errors without displaying alerts, use this function with the Set Error Capture script step with the On option.

Examples

Returns 0 when the most recent script step executed successfully.

Returns 401 when no records are found after the Perform Find script step has been executed.
# Runtime errors

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Unknown error</td>
</tr>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>User canceled action</td>
</tr>
<tr>
<td>2</td>
<td>Memory error</td>
</tr>
<tr>
<td>3</td>
<td>Command is unavailable (for example, wrong operating system, wrong mode, etc.)</td>
</tr>
<tr>
<td>4</td>
<td>Command is unknown</td>
</tr>
<tr>
<td>5</td>
<td>Command is invalid (for example, a Set Field script step does not have a calculation specified)</td>
</tr>
<tr>
<td>6</td>
<td>File is read-only</td>
</tr>
<tr>
<td>7</td>
<td>Running out of memory</td>
</tr>
<tr>
<td>8</td>
<td>Empty result</td>
</tr>
<tr>
<td>9</td>
<td>Insufficient privileges</td>
</tr>
<tr>
<td>10</td>
<td>Requested data is missing</td>
</tr>
<tr>
<td>11</td>
<td>Name is not valid</td>
</tr>
<tr>
<td>12</td>
<td>Name already exists</td>
</tr>
<tr>
<td>13</td>
<td>File or object is in use</td>
</tr>
<tr>
<td>14</td>
<td>Out of range</td>
</tr>
<tr>
<td>15</td>
<td>Can't divide by zero</td>
</tr>
<tr>
<td>16</td>
<td>Operation failed, request retry (for example, a user query)</td>
</tr>
<tr>
<td>17</td>
<td>Attempt to convert foreign character set to UTF-16 failed</td>
</tr>
<tr>
<td>18</td>
<td>Client must provide account information to proceed</td>
</tr>
<tr>
<td>19</td>
<td>String contains characters other than A-Z, a-z, 0-9 (ASCII)</td>
</tr>
<tr>
<td>100</td>
<td>File is missing</td>
</tr>
<tr>
<td>101</td>
<td>Record is missing</td>
</tr>
<tr>
<td>102</td>
<td>Field is missing</td>
</tr>
<tr>
<td>103</td>
<td>Relationship is missing</td>
</tr>
<tr>
<td>104</td>
<td>Script is missing</td>
</tr>
<tr>
<td>105</td>
<td>Layout is missing</td>
</tr>
<tr>
<td>106</td>
<td>Table is missing</td>
</tr>
<tr>
<td>107</td>
<td>Index is missing</td>
</tr>
<tr>
<td>108</td>
<td>Value list is missing</td>
</tr>
<tr>
<td>109</td>
<td>Privilege set is missing</td>
</tr>
<tr>
<td>110</td>
<td>Related tables are missing</td>
</tr>
<tr>
<td>111</td>
<td>Field repetition is invalid</td>
</tr>
<tr>
<td>112</td>
<td>Window is missing</td>
</tr>
<tr>
<td>113</td>
<td>Function is missing</td>
</tr>
<tr>
<td>114</td>
<td>File reference is missing</td>
</tr>
<tr>
<td>Error Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>115</td>
<td>Specified menu set is not present</td>
</tr>
<tr>
<td>116</td>
<td>Specified layout object is not present</td>
</tr>
<tr>
<td>130</td>
<td>Files are damaged or missing and must be reinstalled</td>
</tr>
<tr>
<td>131</td>
<td>Language pack files are missing (such as template files)</td>
</tr>
<tr>
<td>200</td>
<td>Record access is denied</td>
</tr>
<tr>
<td>201</td>
<td>Field cannot be modified</td>
</tr>
<tr>
<td>202</td>
<td>Field access is denied</td>
</tr>
<tr>
<td>203</td>
<td>No records in file to print, or password doesn't allow print access</td>
</tr>
<tr>
<td>204</td>
<td>No access to field(s) in sort order</td>
</tr>
<tr>
<td>205</td>
<td>User does not have access privileges to create new records; import will overwrite existing data</td>
</tr>
<tr>
<td>206</td>
<td>User does not have password change privileges, or file is not modifiable</td>
</tr>
<tr>
<td>207</td>
<td>User does not have sufficient privileges to change database schema, or file is not modifiable</td>
</tr>
<tr>
<td>208</td>
<td>Password does not contain enough characters</td>
</tr>
<tr>
<td>209</td>
<td>New password must be different from existing one</td>
</tr>
<tr>
<td>210</td>
<td>User account is inactive</td>
</tr>
<tr>
<td>211</td>
<td>Password has expired</td>
</tr>
<tr>
<td>212</td>
<td>Invalid user account and/or password. Please try again</td>
</tr>
<tr>
<td>213</td>
<td>User account and/or password does not exist</td>
</tr>
<tr>
<td>214</td>
<td>Too many login attempts</td>
</tr>
<tr>
<td>215</td>
<td>Administrator privileges cannot be duplicated</td>
</tr>
<tr>
<td>216</td>
<td>Guest account cannot be duplicated</td>
</tr>
<tr>
<td>217</td>
<td>User does not have sufficient privileges to modify administrator account</td>
</tr>
<tr>
<td>300</td>
<td>File is locked or in use</td>
</tr>
<tr>
<td>301</td>
<td>Record is in use by another user</td>
</tr>
<tr>
<td>302</td>
<td>Table is in use by another user</td>
</tr>
<tr>
<td>303</td>
<td>Database schema is in use by another user</td>
</tr>
<tr>
<td>304</td>
<td>Layout is in use by another user</td>
</tr>
<tr>
<td>306</td>
<td>Record modification ID does not match</td>
</tr>
<tr>
<td>400</td>
<td>Find criteria are empty</td>
</tr>
<tr>
<td>401</td>
<td>No records match the request</td>
</tr>
<tr>
<td>402</td>
<td>Selected field is not a match field for a lookup</td>
</tr>
<tr>
<td>403</td>
<td>Exceeding maximum record limit for trial version of FileMaker Pro</td>
</tr>
<tr>
<td>404</td>
<td>Sort order is invalid</td>
</tr>
<tr>
<td>405</td>
<td>Number of records specified exceeds number of records that can be omitted</td>
</tr>
<tr>
<td>406</td>
<td>Replace/Reserialize criteria are invalid</td>
</tr>
<tr>
<td>407</td>
<td>One or both match fields are missing (invalid relationship)</td>
</tr>
<tr>
<td>408</td>
<td>Specified field has inappropriate data type for this operation</td>
</tr>
<tr>
<td>Error Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>409</td>
<td>Import order is invalid</td>
</tr>
<tr>
<td>410</td>
<td>Export order is invalid</td>
</tr>
<tr>
<td>412</td>
<td>Wrong version of FileMaker Pro used to recover file</td>
</tr>
<tr>
<td>413</td>
<td>Specified field has inappropriate field type</td>
</tr>
<tr>
<td>414</td>
<td>Layout cannot display the result</td>
</tr>
<tr>
<td>415</td>
<td>One or more required related records are not available</td>
</tr>
<tr>
<td>500</td>
<td>Date value does not meet validation entry options</td>
</tr>
<tr>
<td>501</td>
<td>Time value does not meet validation entry options</td>
</tr>
<tr>
<td>502</td>
<td>Number value does not meet validation entry options</td>
</tr>
<tr>
<td>503</td>
<td>Value in field is not within the range specified in validation entry options</td>
</tr>
<tr>
<td>504</td>
<td>Value in field is not unique as required in validation entry options</td>
</tr>
<tr>
<td>505</td>
<td>Value in field is not an existing value in the database file as required in validation entry options</td>
</tr>
<tr>
<td>506</td>
<td>Value in field is not listed on the value list specified in validation entry option</td>
</tr>
<tr>
<td>507</td>
<td>Value in field failed calculation test of validation entry option</td>
</tr>
<tr>
<td>508</td>
<td>Invalid value entered in Find mode</td>
</tr>
<tr>
<td>509</td>
<td>Field requires a valid value</td>
</tr>
<tr>
<td>510</td>
<td>Related value is empty or unavailable</td>
</tr>
<tr>
<td>511</td>
<td>Value in field exceeds maximum number of allowed characters</td>
</tr>
<tr>
<td>600</td>
<td>Print error has occurred</td>
</tr>
<tr>
<td>601</td>
<td>Combined header and footer exceed one page</td>
</tr>
<tr>
<td>602</td>
<td>Body doesn't fit on a page for current column setup</td>
</tr>
<tr>
<td>603</td>
<td>Print connection lost</td>
</tr>
<tr>
<td>700</td>
<td>File is of the wrong file type for import</td>
</tr>
<tr>
<td>706</td>
<td>EPSF file has no preview image</td>
</tr>
<tr>
<td>707</td>
<td>Graphic translator cannot be found</td>
</tr>
<tr>
<td>708</td>
<td>Can't import the file or need color monitor support to import file</td>
</tr>
<tr>
<td>709</td>
<td>QuickTime movie import failed</td>
</tr>
<tr>
<td>710</td>
<td>Unable to update QuickTime file reference because the database file is read-only</td>
</tr>
<tr>
<td>711</td>
<td>Import translator cannot be found</td>
</tr>
<tr>
<td>714</td>
<td>Password privileges do not allow the operation</td>
</tr>
<tr>
<td>715</td>
<td>Specified Excel worksheet or named range is missing</td>
</tr>
<tr>
<td>716</td>
<td>A SQL query using DELETE, INSERT, or UPDATE is not allowed for ODBC import</td>
</tr>
<tr>
<td>717</td>
<td>There is not enough XML/XSL information to proceed with the import or export</td>
</tr>
<tr>
<td>718</td>
<td>Error in parsing XML file (from Xerces)</td>
</tr>
<tr>
<td>719</td>
<td>Error in transforming XML using XSL (from Xalan)</td>
</tr>
<tr>
<td>720</td>
<td>Error when exporting; intended format does not support repeating fields</td>
</tr>
<tr>
<td>721</td>
<td>Unknown error occurred in the parser or the transformer</td>
</tr>
<tr>
<td>722</td>
<td>Cannot import data into a file that has no fields</td>
</tr>
<tr>
<td>Error Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>723</td>
<td>You do not have permission to add records to or modify records in the target table</td>
</tr>
<tr>
<td>724</td>
<td>You do not have permission to add records to the target table</td>
</tr>
<tr>
<td>725</td>
<td>You do not have permission to modify records in the target table</td>
</tr>
<tr>
<td>726</td>
<td>There are more records in the import file than in the target table. Not all records were imported</td>
</tr>
<tr>
<td>727</td>
<td>There are more records in the target table than in the import file. Not all records were updated</td>
</tr>
<tr>
<td>729</td>
<td>Errors occurred during import. Records could not be imported</td>
</tr>
<tr>
<td>730</td>
<td>Unsupported Excel version. (Convert file to Excel 7.0 (Excel 95), Excel 97, 2000, or XP format and try again)</td>
</tr>
<tr>
<td>731</td>
<td>The file you are importing from contains no data</td>
</tr>
<tr>
<td>732</td>
<td>This file cannot be inserted because it contains other files</td>
</tr>
<tr>
<td>733</td>
<td>A table cannot be imported into itself</td>
</tr>
<tr>
<td>734</td>
<td>This file type cannot be displayed as a picture</td>
</tr>
<tr>
<td>735</td>
<td>This file type cannot be displayed as a picture. It will be inserted and displayed as a file</td>
</tr>
<tr>
<td>736</td>
<td>Too much data to export to this format. It will be truncated</td>
</tr>
<tr>
<td>800</td>
<td>Unable to create file on disk</td>
</tr>
<tr>
<td>801</td>
<td>Unable to create temporary file on System disk</td>
</tr>
<tr>
<td>802</td>
<td>Unable to open file</td>
</tr>
<tr>
<td>803</td>
<td>File is single user or host cannot be found</td>
</tr>
<tr>
<td>804</td>
<td>File cannot be opened as read-only in its current state</td>
</tr>
<tr>
<td>805</td>
<td>File is damaged; use Recover command</td>
</tr>
<tr>
<td>806</td>
<td>File cannot be opened with this version of FileMaker Pro</td>
</tr>
<tr>
<td>807</td>
<td>File is not a FileMaker Pro file or is severely damaged</td>
</tr>
<tr>
<td>808</td>
<td>Cannot open file because access privileges are damaged</td>
</tr>
<tr>
<td>809</td>
<td>Disk/volume is full</td>
</tr>
<tr>
<td>810</td>
<td>Disk/volume is locked</td>
</tr>
<tr>
<td>811</td>
<td>Temporary file cannot be opened as FileMaker Pro file</td>
</tr>
<tr>
<td>813</td>
<td>Record Synchronization error on network</td>
</tr>
<tr>
<td>814</td>
<td>File(s) cannot be opened because maximum number is open</td>
</tr>
<tr>
<td>815</td>
<td>Couldn't open lookup file</td>
</tr>
<tr>
<td>816</td>
<td>Unable to convert file</td>
</tr>
<tr>
<td>817</td>
<td>Unable to open file because it does not belong to this solution</td>
</tr>
<tr>
<td>819</td>
<td>Cannot save a local copy of a remote file</td>
</tr>
<tr>
<td>820</td>
<td>File is in the process of being closed</td>
</tr>
<tr>
<td>821</td>
<td>Host forced a disconnect</td>
</tr>
<tr>
<td>822</td>
<td>FMI files not found; reinstall missing files</td>
</tr>
<tr>
<td>823</td>
<td>Cannot set file to single-user, guests are connected</td>
</tr>
<tr>
<td>824</td>
<td>File is damaged or not a FileMaker file</td>
</tr>
<tr>
<td>Error Number</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>900</td>
<td>General spelling engine error</td>
</tr>
<tr>
<td>901</td>
<td>Main spelling dictionary not installed</td>
</tr>
<tr>
<td>902</td>
<td>Could not launch the Help system</td>
</tr>
<tr>
<td>903</td>
<td>Command cannot be used in a shared file</td>
</tr>
<tr>
<td>905</td>
<td>No active field selected; command can only be used if there is an active field</td>
</tr>
<tr>
<td>920</td>
<td>Can’t initialize the spelling engine</td>
</tr>
<tr>
<td>921</td>
<td>User dictionary cannot be loaded for editing</td>
</tr>
<tr>
<td>922</td>
<td>User dictionary cannot be found</td>
</tr>
<tr>
<td>923</td>
<td>User dictionary is read-only</td>
</tr>
<tr>
<td>951</td>
<td>An unexpected error occurred (*)</td>
</tr>
<tr>
<td>954</td>
<td>Unsupported XML grammar (*)</td>
</tr>
<tr>
<td>955</td>
<td>No database name (*)</td>
</tr>
<tr>
<td>956</td>
<td>Maximum number of database sessions exceeded (*)</td>
</tr>
<tr>
<td>957</td>
<td>Conflicting commands (*)</td>
</tr>
<tr>
<td>958</td>
<td>Parameter missing (*)</td>
</tr>
<tr>
<td>1200</td>
<td>Generic calculation error</td>
</tr>
<tr>
<td>1201</td>
<td>Too few parameters in the function</td>
</tr>
<tr>
<td>1202</td>
<td>Too many parameters in the function</td>
</tr>
<tr>
<td>1203</td>
<td>Unexpected end of calculation</td>
</tr>
<tr>
<td>1204</td>
<td>Number, text constant, field name or “(“ expected</td>
</tr>
<tr>
<td>1205</td>
<td>Comment is not terminated with “*/”</td>
</tr>
<tr>
<td>1206</td>
<td>Text constant must end with a quotation mark</td>
</tr>
<tr>
<td>1207</td>
<td>Unbalanced parenthesis</td>
</tr>
<tr>
<td>1208</td>
<td>Operator missing, function not found or “(“ not expected</td>
</tr>
<tr>
<td>1209</td>
<td>Name (such as field name or layout name) is missing</td>
</tr>
<tr>
<td>1210</td>
<td>Plug-in function has already been registered</td>
</tr>
<tr>
<td>1211</td>
<td>List usage is not allowed in this function</td>
</tr>
<tr>
<td>1212</td>
<td>An operator (for example, +, -, *) is expected here</td>
</tr>
<tr>
<td>1213</td>
<td>This variable has already been defined in the Let function</td>
</tr>
<tr>
<td>1214</td>
<td>AVERAGE, COUNT, EXTEND, GETREPEITION, MAX, MIN, NPV, STDEV, SUM and GETSUMMARY: expression found where a field alone is needed</td>
</tr>
<tr>
<td>1215</td>
<td>This parameter is an invalid Get function parameter</td>
</tr>
<tr>
<td>1216</td>
<td>Only Summary fields allowed as first argument in GETSUMMARY</td>
</tr>
<tr>
<td>1217</td>
<td>Break field is invalid</td>
</tr>
<tr>
<td>1218</td>
<td>Cannot evaluate the number</td>
</tr>
<tr>
<td>1219</td>
<td>A field cannot be used in its own formula</td>
</tr>
<tr>
<td>1220</td>
<td>Field type must be normal or calculated</td>
</tr>
<tr>
<td>1221</td>
<td>Data type must be number, date, time, or timestamp</td>
</tr>
<tr>
<td>1222</td>
<td>Calculation cannot be stored</td>
</tr>
</tbody>
</table>
Get functions

93

Chapter 7  

Error codes marked with an asterisk (*) are returned only by web-published databases.

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1223</td>
<td>The function is not implemented</td>
</tr>
<tr>
<td>1224</td>
<td>The function is not defined</td>
</tr>
<tr>
<td>1300</td>
<td>The specified name can’t be used</td>
</tr>
<tr>
<td>1400</td>
<td>ODBC driver initialization failed; make sure the ODBC drivers are properly installed</td>
</tr>
<tr>
<td>1401</td>
<td>Failed to allocate environment (ODBC)</td>
</tr>
<tr>
<td>1402</td>
<td>Failed to free environment (ODBC)</td>
</tr>
<tr>
<td>1403</td>
<td>Failed to disconnect (ODBC)</td>
</tr>
<tr>
<td>1404</td>
<td>Failed to allocate connection (ODBC)</td>
</tr>
<tr>
<td>1405</td>
<td>Failed to free connection (ODBC)</td>
</tr>
<tr>
<td>1406</td>
<td>Failed check for SQL API (ODBC)</td>
</tr>
<tr>
<td>1407</td>
<td>Failed to allocate statement (ODBC)</td>
</tr>
<tr>
<td>1408</td>
<td>Extended error (ODBC)</td>
</tr>
</tbody>
</table>

Error codes marked with an asterisk (*) are returned only by web-published databases.

**Get(LastMessageChoice)**

**Format**

Get (LastMessageChoice)

**Parameter**

None

**Data type returned**

number

**Description**

Returns a number corresponding to the button clicked in an alert message that is displayed by the Show Custom Dialog script step.

Returns:

- 1 for the first button (by default, labeled OK)
- 2 for the second button (by default, labeled Cancel)
- 3 for the third button

**Note** See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.
Get(LastODBCError)

Format
Get(LastODBCError)

Parameter
None

Data type returned
text

Description
Returns a string that shows the ODBC error state (SQLSTATE), as published by ODBC standards, based on ISO/IEF standards.

You can obtain the error state after an ODBC-related script step has been executed to check for known errors and determine if you want to continue with the script. The ODBC error state is cleared before the next ODBC-related script is executed. An error message, based on the error state returned by the ODBC driver, is displayed.

Notes
• You can set the Set Error Capture state to “on” to suppress the error messages. You can also use Get(LastError) to get generic errors.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns [DataDirect][Macintosh ODBC Driver Manager] Data source name not found and no default driver specified (-1) when a data source name wasn’t found and the driver wasn’t specified.

Get(LayoutAccess)

Format
Get(LayoutAccess)

Parameter
None

Data type returned
number
Description
Returns a number based on record access privileges available through the current layout. You assign the privileges in the Custom Layout Privileges dialog box.

Returns:
• 0 if the custom layout privileges of an account’s privilege set allow no access to Records via this layout.
• 1 if the custom layout privileges of an account’s privilege set allow view only access to Records via this layout. If the database is opened with read-only access, FileMaker Pro returns 1 even if the layout has read-write access privileges.
• 2 if the custom layout privileges of an account’s privilege set allow modifiable access to Records via this layout.

See FileMaker Pro help for more details about limiting access through layouts.

Notes
• Get(LayoutAccess) returns information about record access privileges defined for only the current layout. It ignores current record access privileges for all other layouts. To fully check access through a layout, consider the return values of Get(LayoutAccess) and the Get(RecordAccess) function, page 102.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 1 when the layout allows view-only access to records.

Get(LayoutCount)

Format
Get(LayoutCount)

Parameter
None

Data type returned
number

Description
Returns the total number of layouts in the database file.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.
Example
Returns 3 when the file has three layouts.

Get(LayoutName)

Format
Get(LayoutName)

Parameter
None

Data type returned
text

Description
Returns the name of the layout currently displayed.
If there are multiple windows open in the current database file, each window can have its own layout name value, but results are returned for only the foreground window.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns Product List when the Product List layout is displayed.
Returns Customer Invoice when the Customer Invoice layout is displayed.

Get(LayoutNumber)

Format
Get(LayoutNumber)

Parameter
None

Data type returned
number

Description
Returns the number of the layout currently displayed, according to the list in the Set Layout Order dialog box.
If there are multiple windows open in the current database file, each window can have its own layout number value, but results are returned for only the foreground window.
Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 3 when the current layout is third in the list of layouts in Set Layout Order.

Get(LayoutTableName)

Format
Get(LayoutTableName)

Parameter
None

Data type returned
text

Description
Returns the name of the table from which the current layout is displaying records. If no windows are open, an empty string is returned.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
There are two layouts, Teachers Layout and Coaches Layout, with corresponding tables named Teachers and Coaches in the table Instructors. An unstored calculation of Get(LayoutTableName) returns Teachers when the current layout is Teachers Layout and returns Coaches when the current layout is Coaches Layout.

Get(LayoutViewState)

Format
Get(LayoutViewState)

Parameter
None

Data type returned
number

Description
Returns a number indicating the currently active database file view. Returns:
• 0 (zero) if the database file is in View as Form view
• 1 if the database file is in View as List view
• 2 if the database file is in View as Table view

If there are multiple windows open in the current database file, each window can have its own layout view state value, but results are returned for only the foreground window.

**Note** See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

### Get(MultiUserState)

**Format**

Get(MultiUserState)

**Parameter**

None

**Data type returned**

number

**Description**

Returns a number representing the level of sharing for the database file using FileMaker Network.

Returns:

• 0 when network sharing is off.

• 1 when network sharing is on, you’re accessing the database file from the host computer, and either all users or a specific group of users (based on their privilege set) have network access to the database file.

• 2 when network sharing is on, you’re accessing the database file from a client computer, and either all users or a specific group of users (based on their privilege set) have network access to the database file.

**Notes**

• If the current calculation is stored and you specify its context, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.

• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Example**

Returns 0 when access is denied to other users.
Get(NetworkProtocol)

Format
Get(NetworkProtocol)

Parameter
None

Data type returned
text

Description
Returns the name of the network protocol (TCP/IP) that FileMaker Pro is using on this machine.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns TCP/IP.

Get(PageNumber)

Format
Get(PageNumber)

Parameter
None

Data type returned
number

Description
Returns a number representing the current page being printed or previewed. If nothing is being printed or previewed, 0 is returned.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 4 when page 4 is being printed or previewed.
Get(PortalRowNumber)

Format
Get(PortalRowNumber)

Parameter
None

Data type returned
number

Description
Returns the number of the currently selected portal row. When no portal row is selected, returns 0. If there are multiple windows open in the current database file, each window can have its own portal row number value, but results are returned for only the foreground window.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 5 when the fifth row of a portal is currently selected, or when the cursor is in a field in the fifth portal row.

Get(PreferencesPath)

Format
Get(PreferencesPath)

Parameter
None

Data type returned
text

Description
Returns the path to the preferences and default options folder for the current user. In Windows, the path format is /Drive:/Documents and Settings/UserName/Local Settings/Application Data/. In the Mac OS, the path format is /DriveName/Users/UserName/Library/Preferences/.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.
Examples
Returns /C:/Documents and Settings/John Smith/Local Settings/Application Data/ for a user named John Smith in Windows.
Returns /MacintoshHD/Users/John Smith/Library/Preferences/ for a user named John Smith in the Mac OS.

Get(PrinterName)

Format
Get(PrinterName)

Parameter
None

Data type returned
text

Description
In Windows, returns a string with each of these entries separated by a comma:
• the printer name
• the driver name
• the name of the printer port
In Mac OS, returns a string with these entries separated by the word on:
• the Queue name of the printer (if provided)
• the IP address of the printer
If any of this information isn’t available, <Unknown> is inserted in the result (except for Queue name in the Mac OS).

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
For Windows:
Returns HP LaserJet 4, WINSPOOL, LPT1.
For Mac OS:
Returns 24.109.265.43.
Get(PrivilegeSetName)

**Format**
Get(PrivilegeSetName)

**Parameter**
None

**Data type returned**
text

**Description**
Returns the name of the privilege set assigned to the account being used by the current user of the database file. If a user is using the default Admin account and you haven’t modified access privileges for the database file, Get(PrivilegeSetName) returns [Full Access].

**Notes**
- If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
- See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Examples**
For current user Administrator, Get(PrivilegeSetName) might return [Full Access].
For a user in the sales department, Get(PrivilegeSetName) might return [Data Entry Only].

Get(RecordAccess)

**Format**
Get(RecordAccess)

**Parameter**
None

**Data type returned**
number
Description

Returns a number based on the current record’s access privileges, assigned through the Custom Record Privileges dialog box.

Returns:

- 0 if the custom record privileges of an account’s privilege set have neither View nor Edit privileges set to yes for the current record.
- 1 if the custom record privileges of an account’s privilege set have View set to yes for the current record, or if View is set to limited and the calculation defined for limited access returns a value of true.
- 2 if the custom record privileges of an account’s privilege set have Edit set to yes for the current record, or if Edit is set to limited and the calculation defined for limited access returns a value of true.

Note If both View and Edit are set to yes, Get(RecordAccess) returns 2.

See FileMaker Pro help for more details about limiting access to records.

Notes

- Get(RecordAccess) only returns information about the privileges defined for accessing records. It ignores access privileges assigned through individual layouts. To fully check access to a record, consider the return values of the Get(LayoutAccess) function, page 94, and Get(RecordAccess).
- If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
- See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example

Returns 1 when the record access is view-only.

Get(RecordID)

Format

Get (RecordID)

Parameter

None

Data type returned

number
Description
Returns the unique ID number of the current record. This number is a decimal value (an integer) generated by FileMaker Pro when the record is created. It does not change.

Notes
• If the current calculation is stored and you specify its context, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns a unique ID for the current record.

Get(RecordModificationCount)

Format
Get(RecordModificationCount)

Parameter
None

Data type returned
number

Description
Returns the total number of times changes to the current record have been committed. To commit changes, you can, for example:
• click out of all fields (exit the record)
• go to a different record
• enter Find mode
If multiple windows are open, clicking in another window does not commit the record.

Notes
• If the current calculation is stored and you specify its context, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.
Example
Returns 0 if the record has not been modified since it was created.
If changes are made to four fields and all four fields are committed together, the result increments by one. If changes are made to four fields and each change is committed separately, the result increments by four.

Get(RecordNumber)

Format
Get (RecordNumber)

Parameter
None

Data type returned
number

Description
Returns the number of the current record in the current found set. This value is determined by the relative place of the record in the found set, and changes depending on the find criteria and the sort order.

Notes
• To return a value that uniquely and permanently identifies a record in this table, use Get (RecordID).
• If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 3 when the current record is the third record in a found set.

Get(RecordOpenCount)

Format
Get (RecordOpenCount)

Parameter
None
Data type returned
number

Description
Returns the total number of open records in the current found set that haven’t been saved.

Notes
• If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 4 if there are four open records in the current found set that haven’t been saved.

Get(RecordOpenState)

Format
Get (RecordOpenState)

Parameter
None

Data type returned
number

Description
Returns a number representing the state of the current record. Returns:
• 0 for a closed or committed record.
• 1 for a new record that hasn’t been committed.
• 2 for a modified record that hasn’t been committed.

Notes
• If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.
Example
Returns 1 if the current record is a new record that hasn’t been saved.

Get(RequestCount)

Format
Get (RequestCount)

Parameter
None

Data type returned
number

Description
Returns the total number of find requests defined for the current table. If there are multiple windows open in the current database file, then results are returned for only the top-most window of the file that the calculation is defined in.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 5 when there are five find requests defined for the current table.

Get(RequestOmitState)

Format
Get (RequestOmitState)

Parameter
None

Data type returned
number

Description
Returns a Boolean value representing the state of the Omit checkbox in Find mode. Returns 1 if the Omit checkbox is selected, otherwise returns 0.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.
Example
Returns 1 when the Omit checkbox is selected in the current find request.

Get(ScreenDepth)

Format
Get(ScreenDepth)

Parameter
None

Data type returned
number

Description
Returns the number of bits needed to represent the color or shade of gray of a pixel on the main screen. A value of 8 represents 256 (equal to $2^8$) colors or shades of gray.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns 32 on a display showing millions ($2^{32}$) of colors.
Returns 16 on a display showing thousands ($2^{16}$) of colors.
Returns 4 on a VGA display.
Returns 1 on a black-and-white display.

Get(ScreenHeight)

Format
Get(ScreenHeight)

Parameter
None

Data type returned
number
Description
Returns the number of pixels displayed vertically on the screen in which the window of the current file is open. When the window spans more than one screen, this function uses the screen that contains the largest percentage of the window. If there are multiple windows open in the current database file, each window can have its own screen height value, but results are returned for only the foreground window.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 480 when the screen resolution is set to 640 by 480.

Get(ScreenWidth)

Format
Get(ScreenWidth)

Parameter
None

Data type returned
number

Description
Returns the number of pixels displayed horizontally on the screen in which the window of the current file is open. When the window spans more than one screen, this function uses the screen that contains the largest percentage of the window. If there are multiple windows open in the current database file, each window can have its own screen width value, but results are returned for only the foreground window.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 640 when the screen resolution is set to 640 by 480.

Get(ScriptName)

Format
Get(ScriptName)

Parameter
None
Data type returned
text

Description
Returns the name of the script currently running (or paused).

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns Print Report when the Print Report script is running.
Returns Update Customer when the Update Customer script is running.

Get(ScriptParameter)

Format
Get(ScriptParameter)

Parameter
None

Data type returned
text

Description
When this function is part of a calculation evaluated within a script, returns the script parameter passed into the script.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns “Print” when “Print” was the value of the parameter passed into the current script.
The following example shows how to pass named parameters using the Evaluate, Let, and Get(ScriptParameter) functions, allowing access only to variable 'a' (the example returns 6):

```
ScriptParameter = "a = 5; b = 10"
Evaluate("Let ( [" & Get(ScriptParameter) & "]); a + 1 )")
```

The following example shows how to pass named parameters, allowing access to both variable 'a' and 'b'. The simplified first parameter makes the second parameter more complex (the example returns 6, 12):

```
ScriptParameter = "a = 5; b = 10"
```
Evaluate("Let ( [" & Get(ScriptParameter) & "] ; a + 1 & ", " & b + 2 )")

The following example shows how to pass named parameters, while keeping the ability to check the syntax of the second parameter of the Let function (the example returns 6, 12):

ScriptParameter = "a = 5; b = 10"
Let( [a = Evaluate("Let( [" & Get(ScriptParameter) & "] ; a )")], b = Evaluate("Let( [" & Get(ScriptParameter) & "] ; b )")]; a + 1 & ", " & b + 2 )

Get(ScriptResult)

Format
Get(ScriptResult)

Parameter
None

Data type returned
text, number, date, time, timestamp, container

Description
When this function is part of a calculation evaluated within a script, returns the script result from a performed subscript. If a subscript doesn’t return a result, then the content of the script result will be empty.

Example
In the following example, script Find Customers returns the results of a find request when it is called from script Do Reports. Script Find Customers uses the optional script result of the Exit Script script step. Script Do Reports then uses Get(ScriptResult) to determine what other script steps should be performed based on the returned result stored in Get(ScriptResult).

Find Customers
Set Error Capture [On]
Perform Find [Restore]
New Record/Request
Exit Script [Result: Get(FoundCount) < 10]

Do Reports
Perform Script [Find Customers]
If [Get(ScriptResult) = 0]
    Show Custom Dialog [“You have created 10 records already.”]
End If
Get(SortState)

Format
Get(SortState)

Parameter
None

Data type returned
number

Description
Returns 0 if the records in the active table are not sorted.
Returns 1 if the records in the active table are sorted.
Returns 2 if the records in the active table are partially sorted (semi-sorted).
Each window has its own sort state.

Notes
- The records in a sorted table can become semi-sorted if you add a new record, or if you change the contents of a field that was involved in the sorting.
- If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 1 when the records in the active table are sorted.

Get(StatusAreaState)

Format
Get(StatusAreaState)

Parameter
None

Data type returned
number
Description
Returns a number indicating the current status area state.

Returns:
- 0 (zero) if the status area is hidden
- 1 if the status area is visible
- 2 if the status area is visible and locked
- 3 if the status area is hidden and locked

If there are multiple windows open on the currently active database file, then results are returned for only the active window.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 1, when the current status area is visible.

Get(SystemDrive)

Format
Get(SystemDrive)

Parameter
None

Data type returned
text

Description
Returns the drive letter (Windows) or volume name (Mac OS) where the currently running operating system is located.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns /C:/ in Windows when the operating system is on the C: drive.
Returns /DriveName/ in the Mac OS when the operating system is on a volume named DriveName.
Get(SystemIPAddress)

Format
Get(SystemIPAddress)

Parameter
None

Data type returned
text

Description
Returns a list of the IP addresses of all the machines connected to a NIC (Network Interface Controller) card. IP addresses are separated by carriage returns.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 14.156.13.121, for example, when only one machine is connected.

Get(SystemLanguage)

Format
Get(SystemLanguage)

Parameter
None

Data type returned
text

Description
Returns the language currently set on the current system. The text that is returned is in the English language.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns Japanese when Japanese is the language currently set on the operating system.
Get(SystemNICAddress)

**Format**
Get(SystemNICAddress)

**Parameter**
None

**Data type returned**
text

**Description**
Returns the hardware address of all the Network Interface Controller cards connected to the machine. The address consists of 6 bytes displayed in hexadecimal separated by colons. In Windows, find this address by typing the command “ipconfig /All” in a DOS window. In the Mac OS, find this address under Network Overview in the System Profile tab under Applications/Utilities/Apple System Profiler.

**Note** See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Example**
Returns 00:07:34:4e:c2:0d, for example.

Get(SystemPlatform)

**Format**
Get(SystemPlatform)

**Parameter**
None

**Data type returned**
number

**Description**
Returns a number indicating the current platform:
- -1 if the current platform is Mac OS X
- -2 if the platform is Windows 2000 or Windows XP

**Note** See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.
Examples

Abs(Get(SystemPlatform)) returns 2 when the current platform is a Windows platform.

Get(SystemPlatform) returns -1 when the current platform is Mac OS X.

Get(SystemVersion)

Format
Get(SystemVersion)

Parameter
None

Data type returned
text

Description
When this function is used in a script, returns the version of the operating system of the machine of the person running the script.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples

Returns 5.0 for Windows 2000.
Returns 5.1 for Windows XP.
Returns 10.3 for Mac OS X version 10.3.

Get(TextRulerVisible)

Format
Get(TextRulerVisible)

Parameter
None

Data type returned
number

Description
Returns a Boolean value representing whether or not the text ruler is visible. Returns 1 if the text ruler is displayed, otherwise returns 0.
Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns 1 when the text ruler is visible.

Get(TotalRecordCount)

Format
Get (TotalRecordCount)

Parameter
None

Data type returned
number

Description
Returns the total number of records in the current table.

Notes
• If the current calculation is stored and you specify its context, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 876 when there are 876 records in the current table.

Get(UserCount)

Format
Get (UserCount)

Parameter
None

Data type returned
number
Description
Returns the number of clients currently accessing the file.
Returns:
• 1 if FileMaker network sharing is turned off
• 1 + the number of clients if FileMaker network sharing is turned on
This function does not count clients accessing the database file via the web, ODBC or JDBC.

Notes
• If you specify the context for the current calculation, this function will be evaluated based on that context; otherwise, it will be evaluated based on the context of the current window.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 5 when there are 4 clients accessing the database file.

Get(UserName)

Format
Get(UserName)

Parameter
None

Data type returned
text

Description
Returns the name of the FileMaker Pro user, as specified in the General tab of the Preferences dialog box. The returned name is user-specified.

Important For greater security, use Get(AccountName) to track and manage user access: a user cannot change the account name used to log in to a database file.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns Sharon Lloyd when Sharon Lloyd is the current user.
Get(UseSystemFormatsState)

Format
Get(UseSystemFormatsState)

Parameter
None

Data type returned
number

Description
Returns a Boolean value representing the state of the Use System Formats command in the Format menu. Returns 1 if Use System Formats is on, otherwise returns 0.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns 1 when Use System Formats is on.

Get(WindowContentHeight)

Format
Get(WindowContentHeight)

Parameter
None

Data type returned
number

Description
Returns a number representing the height, in pixels, of the FileMaker Pro content area. The content area depends on the current size of the active window but doesn't include the title bar, scroll bars, zoom controls, and page margins. The content area is the space inside these controls.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns 400 in the Mac OS when the current window height is 437.
The example below combines \texttt{Get(WindowContentHeight)} with \texttt{Get(WindowHeight)} to determine the size of the title bar and horizontal scroll bar:

\[
\text{Get(WindowHeight)} - \text{Get(WindowContentHeight)} \quad \text{returns 37 in the Mac OS when the window height is 437.}
\]

\section*{Get(WindowContentWidth)}

\textbf{Format}

\texttt{Get(WindowContentWidth)}

\textbf{Parameter}

None

\textbf{Data type returned}

number

\textbf{Description}

Returns a number representing the width, in pixels, of the FileMaker Pro content area. The content area depends on the current size of the active window but doesn’t include the title bar, scroll bars, zoom controls, or page margins. It does include the status area if it is currently showing. The content area is the space inside these controls.

\textbf{Note} See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

\textbf{Examples}

Returns 400 in the Mac OS when the current window width is 415 and the status area isn’t showing.

The example below combines \texttt{Get(WindowContentWidth)} with \texttt{Get(WindowWidth)} to determine the size of the status bar and vertical scroll bar:

\[
\text{Get(WindowWidth)} - \text{Get(WindowContentWidth)} \quad \text{returns 15 in the Mac OS when the window width is 415 and the status area isn’t showing.}
\]

\section*{Get(WindowDesktopHeight)}

\textbf{Format}

\texttt{Get(WindowDesktopHeight)}

\textbf{Parameter}

None

\textbf{Data type returned}

number
Description
Returns a number representing the height, in pixels, of the desktop space.
In Windows, the desktop space is the area inside the MDI window (sometimes referred to as the client area). This doesn’t include any virtual space available through the scrolling of the MDI window.
In the Mac OS, the desktop space is the area on the main monitor excluding the menu bars and Dock. The main monitor is where the menu bar is located.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns 460 in Windows when there is a single monitor and its MDI is set to 500 x 450.
Returns 578 in the Mac OS when there is a single monitor and its resolution is set to 800 x 600.

Get(WindowDesktopWidth)

Format
Get(WindowDesktopWidth)

Parameter
None

Data type returned
number

Description
Returns a number representing the width, in pixels, of the desktop space.
In Windows, the desktop space is the space inside the MDI window (sometimes referred to as the client area).
In the Mac OS, the desktop space is the area on the main monitor excluding the menu bars and Dock. The main monitor is where the menu bar is located.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns 450 in Windows when there is a single monitor and its MDI is set to 500 x 450.
Returns 600 in the Mac OS when there is a single monitor and its resolution is set to 800 x 600.
Get(WindowHeight)

Format
Get(WindowHeight)

Parameter
None

Data type returned
number

Description
Returns a number representing the height, in pixels, of the window that the script is acting on (not necessarily the foreground window). The height of the window is calculated from the top to bottom outer edges of the window. This position doesn't include shadows or other effects applied to windows.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Get(WindowHeight) returns 300 when the current window’s height is 300 pixels.

Get(WindowLeft)

Format
Get(WindowLeft)

Parameter
None

Data type returned
number

Description
Returns a number representing the horizontal distance, in pixels, of the outer edge of the window that the script is acting on (not necessarily the foreground window) relative to the left-most edge of the screen. The origin of the reference coordinate system is at the left-most corner below the menu bar or toolbar. A negative value indicates the portion of the left side of the window that is hidden.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.
Examples
Returns 52 when the outer edge of the active window is 52 pixels from the left edge of the screen.
Returns 0 when the active window is 0 pixels from the left edge of the screen.

Get(WindowMode)

Format
Get (WindowMode)

Parameter
None

Data type returned
number

Description
Returns a number representing the mode FileMaker Pro is in at the time the function is evaluated:
• 0 for Browse mode
• 1 for Find mode
• 2 for Preview mode
• 3 if printing is in progress

If a script using this function runs while the file is in Layout mode, FileMaker Pro switches to Browse mode and returns 0. If there are multiple windows open in the current database file, each window can have its own window mode value, but results are returned for only the foreground window.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 2 if the file is in Preview mode when the function is evaluated.

Get(WindowName)

Format
Get (WindowName)

Parameter
None
Data type returned
text

Description
Returns the name of the window that the script is acting on (not necessarily the foreground window). Returns an empty string if there is no window.

Notes
• You can set the window name with the Set Window Title script step.
• See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
There are two windows, Teachers and Students, displaying the same layout that includes an unstored calculation Calc containing Get(WindowName). Teachers is returned when the Teachers window is refreshed, and Students is returned when the Students window is refreshed.

Get(WindowTop)

Format
Get(WindowTop)

Parameter
None

Data type returned
number

Description
Returns a number representing the vertical distance, in pixels, of the outer edge of the window that the script is acting on (not necessarily the foreground window) relative to the bottom edge of the menu bar or toolbar. The origin of the reference coordinate system is at the left-most corner below the menu bar or toolbar. A negative value indicates the portion of the top part of the window that is hidden behind the menu bar or toolbar.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns 52 when the outer edge of the active window is 52 pixels from the menu bar or toolbar.

Returns 0 when the outer edge of the active window just touches the menu bar or toolbar.
Get(WindowVisible)

**Format**
Get(WindowVisible)

**Parameter**
None

**Data type returned**
number

**Description**
Returns a number representing whether or not the current window is visible. The current window is the window that the script is acting on (not necessarily the foreground window). Returns a 1 if the window is visible. Returns a 0 if the window is hidden using the Hide Window command. The window can be located outside of the visible screen space and still return 1.

**Note** See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

**Examples**
Returns 1 when the current window is physically visible.

Returns 0 when the current window has been hidden using the FileMaker Pro Hide Window command.

Get(WindowWidth)

**Format**
Get(WindowWidth)

**Parameter**
None

**Data type returned**
number

**Description**
Returns a number representing the width, in pixels, of the window that the script is acting on (not necessarily the foreground window). The width of the window is calculated from the left-most to right-most outer edge of the window. This position doesn’t include shadows or other effects applied to windows.
Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Example
Returns 300 when the current window is 300 pixels wide.

Get(WindowZoomLevel)

Format
Get (WindowZoomLevel)

Parameter
None

Data type returned
text

Description
Returns the zoom percentage of the current window.
In Windows, an asterisk appears next to the zoom percentage when Enlarge window contents to improve readability is selected in the General tab of the Preferences dialog box.

Note See FileMaker Pro help for information about running scripts in client/server and peer-to-peer environments.

Examples
Returns 200 when the current window’s zoom percentage is set to 200.
Returns 200* in Windows when the current window’s zoom percentage is set to 200 and Enlarge window contents to improve readability is selected.
Chapter 8  
Logical functions

Logical functions test for a condition to evaluate it as true or false. This is known as a Boolean value. If the condition is true, FileMaker Pro returns a 1; if the condition is false, FileMaker Pro returns a 0. You can use the keywords True and False with logical functions and operators when a Boolean value is needed. Keyword True returns 1 and keyword False returns 0.

Logical functions can also evaluate parameters such as text or arithmetic operations that do not make a true or false statement, or in the case of the GetField function, return the contents of another field.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case, page 128</td>
<td>One of several possible results based on a series of tests.</td>
</tr>
<tr>
<td>Choose, page 128</td>
<td>One result value, according to the integer value of a specified test.</td>
</tr>
<tr>
<td>Evaluate, page 129</td>
<td>Evaluates the specified expression as a calculation.</td>
</tr>
<tr>
<td>EvaluationError, page 131</td>
<td>An error code, if any, from the specified expression.</td>
</tr>
<tr>
<td>GetAsBoolean, page 131</td>
<td>0 if the specified data has a value of 0 or is empty, otherwise returns 1.</td>
</tr>
<tr>
<td>GetField, page 132</td>
<td>The contents of the referenced field.</td>
</tr>
<tr>
<td>GetLayoutObjectAttribute, page 133</td>
<td>The requested layout object attributes from the calculation's active window.</td>
</tr>
<tr>
<td>GetNthRecord, page 134</td>
<td>The contents of the referenced field from the requested record number.</td>
</tr>
<tr>
<td>If, page 135</td>
<td>One of two possible results depending on the value of the specified test.</td>
</tr>
<tr>
<td>IsEmpty, page 136</td>
<td>True(1) if the specified field is empty, if the related field, related table, relationship, or file is missing, or if some other error occurs; otherwise returns False(0).</td>
</tr>
<tr>
<td>IsValid, page 137</td>
<td>0 when a record contains an invalid value because of a data type mismatch (text in a date field, for example).</td>
</tr>
<tr>
<td>IsValidExpression, page 137</td>
<td>True(1) if the syntax of the specified expression is correct.</td>
</tr>
<tr>
<td>Let, page 138</td>
<td>Sets variable to the result of value for the duration of the specified expression.</td>
</tr>
<tr>
<td>Lookup, page 140</td>
<td>The value specified in the sourceField parameter using the relationships in the relationships graph.</td>
</tr>
<tr>
<td>LookupNext, page 141</td>
<td>The value specified in the sourceField parameter using the relationships in the relationships graph.</td>
</tr>
</tbody>
</table>
Case

Format
Case(test1;result1{;test2;result2;defaultResult...})

Parameters
- test - any text or numeric expression.
- result - result corresponding to the expression.

Data type returned
text, number, date, time, timestamp, container

Description
Returns one of several possible results based on a series of tests. The Case function evaluates each test expression in order, and when a True expression is found, returns the value specified in result for that expression.

You can include a default result at the end of the parameter list. If none of the expressions evaluate to True, the Case function returns the value specified for defaultResult. If no default result is supplied, the Case function returns an “empty” result.

Examples
Case(Score >= 90;"Excellent";Score > 50;"Satisfactory";"Needs Improvement") displays Excellent when the score is 90 or above, Satisfactory when the score is between 50 and 90, and Needs Improvement for any other score.

Case(Shipment Method="Ground";2;Shipment Method="Air";10) returns 10, when the Shipment Method field contains Air.

Choose

Format
Choose(test;result0{;result1;result2...})

Parameters
- test - Any integer calculation. The calculation result of test must be a number that indexes into the list that follows. Because the index is a 0 based index, the test result must be 0 to access the first result.
- result - one or more results.

Parameters in curly braces {} are optional.
Data type returned
text, number, date, time, timestamp, container

Description
Returns one result value, according to the integer value of test. FileMaker Pro evaluates test to obtain an index number, which is used to choose the corresponding ordinal result.

Because the Choose function is a 0 based list, the first item on the list is indexed 0 and the second item on the list is indexed 1. For example, if test evaluates to 2, then result2 is chosen.

Example
Choose(Rating;“Not Applicable”;“Good”;“Fair”;“Poor”)

Rating is a number field that is empty or holds a value between 1 and 3. If Rating is empty, the Choose function returns nothing. If Rating is 1, the result is Good. If Rating is 2, the result is Fair, and if it is 3, the result is Poor.

Evaluate

Format
Evaluate(expression{;[field1;field2;field3;...]})

Parameters
expression - any text expression or text field.
fields - a list of fields that this function is dependent on. When these fields are modified, the calculation will update its result.

Parameters in curly braces {} are optional. Notice that the optional field list is enclosed in square brackets [ ].

Data type returned
text, number, date, time, timestamp, container

Description
Evaluates expression as a calculation.

The optional fields parameter is a list of fields this calculation is dependent on. If a necessary field isn’t listed, modifying that dependent field won’t update the result of the calculation.

Examples
Evaluate(TextField) returns 4 when TextField contains 2 + 2.
Evaluate("textfield") returns 2 + 2 when textfield contains 2 + 2.
Evaluate(GetField("textfield")) returns 4 when textfield contains 2 + 2.
Evaluate(TextField; [Amount]) returns .80 when TextField contains .08 * Amount and the Amount field contains 10.00.
Evaluate("Let(TaxRate=.05;"& Tax Rate Calculation &")") returns .50 when the field Tax Rate Calculation contains SubTotal * TaxRate where SubTotal is a numeric field that contains 10.00.

The following example shows how to pass named parameters using the Evaluate, Let, and Get(ScriptParameter) functions, allowing access only to variable 'a' (the example returns 6):

ScriptParameter = "a = 5; b = 10"
Evaluate("Let ( [" & Get(ScriptParameter) & "]; a + 1 )")

The following example shows how to pass named parameters, allowing access to both variable 'a' and 'b'. The simplified first parameter makes the second parameter more complex (the example returns 6, 12):

ScriptParameter = "a = 5; b = 10"
Evaluate("Let ( [" & Get(ScriptParameter) & "]; a + 1 & ", " & b + 2 )")

The following example shows how to pass named parameters, while keeping the ability to check the syntax of the second parameter of the Let function (the example returns 6, 12):

ScriptParameter = "a = 5; b = 10"
Let( [a = Evaluate("Let( [" & Get(ScriptParameter) & "]; a )"),
    b = Evaluate("Let( [" & Get(ScriptParameter) & "]; b )")];
    a + 1 & ", " & b + 2 )

Note The Evaluate function evaluates an expression, including field values to be evaluated as a calculation formula. It also allows you to specify field dependencies so that a calculation using the evaluation function can be triggered due to changes in other fields of the same record. This function evaluates user-defined formulas. For example, you can create a formula in the Total field that computes state tax:

Evaluate(StateTaxFormula) + ShippingCost

where the StateTaxFormula field contains:

SubTotal * 1.0875

and the SubTotal field contains the subtotal before tax and shipping.

The Evaluate function has an optional second parameter, which is a field the calculation is dependent on. When the dependent field contents change, FileMaker Pro re-evaluates the calculation. In the following example, the Total calculation will be re-evaluated when SubTotal changes:

Evaluate(StateTaxFormula, SubTotal) + ShippingCost
The dependent parameter can also be useful in other cases. For example, `Evaluate("Get(CurrentTimeStamp)", [FieldB, FieldC])` will store a timestamp in the calculation field whenever FieldB or FieldC changes.

**EvaluationError**

**Format**

`EvaluationError(expression)`

**Parameter**

expression - any calculation `expression`

**Data type returned**

number

**Description**

Returns an error code, if any, from `expression`. There are two types of errors: syntax errors and runtime errors. A syntax error indicates an invalid calculation. A runtime error, such as Field missing or Record missing, occurs when the calculation currently being run is valid but cannot properly execute. See Get(LastError) function, page 87, for a list of error codes and messages.

**Note** The `EvaluationError` function must enclose the `Evaluate` function to return any syntax errors.

**Examples**

`EvaluationError(calculationField)` returns 102 (Field Missing) when `calculationField` contains `total + 1` and the field `total` has been deleted or renamed.

`EvaluationError(Evaluate(calculationField))` returns 1207 (Unbalanced Parenthesis) when `calculationField` contains `abs(-1` with no closing parenthesis.

**GetAsBoolean**

**Format**

`GetAsBoolean(data)`

**Parameter**

data - any text, number, date, time, timestamp or container `expression`, or a field containing text, a number, date, time, timestamp or container.

**Data type returned**

number
Description
Returns 0 if data has a value of 0 or is empty, all other values return 1.

Examples
GetAsBoolean(““) returns 0.
GetAsBoolean(“Some text here.“) returns 0.
GetAsBoolean(Container Field) returns 1 when the field named Container Field contains data, or returns 0 when Container Field is empty.

GetField

Format
GetField(fieldName)

Parameter
fieldName - any text expression or text field that refers to a field’s name

Important See chapter 4, “Design functions” for information about literal text parameters.

Data type returned
text, number, date, time, timestamp, container

Description
Returns the contents of fieldName. Use this function to get fieldName’s contents, or in any function that uses a field, such as NPV, GetSummary, GetRepetition, or the aggregate functions.

Examples
Assume you have the following fields: Arrow and Target. Arrow contains the text string Target, and Target contains the text string Bullseye.

• GetField(“Arrow”) returns Target. Notice the use of quotation marks around Arrow to indicate the literal string is the fieldName parameter.

• GetField(Arrow) returns Bullseye. Notice the absence of quotation marks to indicate the value stored in the Arrow field is the fieldName parameter.

Assume you have the following two fields: FirstName and LastName. FirstName contains the text string Jane, and LastName contains the text string Public.

• GetField(“FirstName”)&” “&GetField(“LastName”) returns the text string Jane Public.

GetSummary(GetField(“Field1”), GetField(“Field” & “2”) performs a summary on the summary field Field1, using a break field of Field2.
GetLayoutObjectAttribute

Format
GetLayoutObjectAttribute(objectName;attributeName{;repetition Number})

Parameters
objectName - the name of a named layout object on the current layout
attributeName - the name of a supported attribute (see below)
repetitionNumber - the repetition number (for repeating fields)
Parameters in curly braces {} are optional.

Data type returned
text

Description
Returns the specified attributes of the layout object given by objectName that is currently active in the calculation.

Attributes:
objectType - returns the object's type as a text literal, in English. Valid return values are: field, text, graphic, line, rectangle, rounded rectangle, oval, group, button group, portal, tab panel, web viewer, and unknown.
hasFocus - returns 1 (True) if objectName is currently active, otherwise returns 0 (False). Objects that can have the focus are fields, portals, tab panels, and groups. Also returns 1 for a portal when a portal row is selected.
containsFocus - returns 1 (True) if objectName is currently active or if it contains an active object; otherwise returns 0 (False). Objects that can contain the focus are fields, portals, tab panels, and groups.
isFrontTabPanel - returns 1 (True) if the target object is the tab panel that is in front.
The following object coordinates are given in pixels relative to the bottom-left corner of the FileMaker menu bar:
bounds - returns a list of numeric values, separated by spaces, that describes the placement of the specified object (left-top to right-bottom).
left - returns the left edge coordinate of the specified object.
right - returns the right edge coordinate of the specified object.
top - returns the top edge coordinate of the specified object.
bottom - returns the bottom edge coordinate of the specified object.
width - returns a number representing the width (in pixels) of the specified object.
height - returns a number representing the height (in pixels) of the specified object.
rotation - returns a number representing the rotation (in degrees) of the specified object.
startPoint, endPoint - returns a pair of numeric values (horizontal vertical), separated by spaces, that represent the start point or end point of a line object. Other objects will return the top left point for startPoint and the bottom right point for endPoint.
source - returns the source description of the specified object as follows. For:
• web viewers - returns current URL
• fields - returns the fully qualified field name (table name::field name)
• text objects - returns the text (does not return merge fields)
• portals - returns the related table name
• graphics - returns image data such as Container data type
For all other objects, returns an empty string.
content - returns the content of the specified object as follows. For:
• web viewers - returns the current content (such as HTML code)
• fields - returns the field data formatted using the specified object's properties
• text objects - returns the text (including text from merge fields)
• graphics - returns image data such as Container data type
For all other objects, returns an empty string.
enclosingObject - returns objectName of the enclosing layout object. Otherwise, returns an empty string. Only groups, tab panels, and portals can contain other objects.
containedObjects - returns a list of named objects contained within objectName. Only groups, tab panels, and portals can contain other objects.

Note The hasFocus, containsFocus, source, and content attributes behave differently in Instant Web Publishing. For more information, see the FileMaker Instant Web Publishing Guide, which is located in the folder where FileMaker Pro is installed.

Examples
GetLayoutObjectAttribute("CancelButton","objectType") returns group
GetLayoutObjectAttribute("CancelButton","bounds") returns 138 24 391 38 0

GetNthRecord

Format
GetNthRecord(fieldName;recordNumber)
Parameters

fieldName - any related field or repeating field, or an expression that returns a field or a repeating field.

recordNumber - the record number from which you want data.

Data type returned
text, number, date, time, timestamp, container

Description

Returns the contents of fieldName from the provided recordNumber.

Note The result of GetNthRecord() will not be updated when the record referred to by GetNthRecord() is a record other than the one in which the calculation is currently being evaluated.

GetNthRecord of the current table returns the Nth record of the found set according to how the current table is sorted.

GetNthRecord of a related table returns the Nth record of the related set (relative to the current record), regardless of how the related table (or portal) is sorted.

Examples

GetNthRecord(First Name;2) returns the contents of the First Name field for record 2 in the current table.

GetNthRecord(First Name;Get(RecordNumber)+ 1) returns the contents of the First Name field for the next record in the current table.

GetNthRecord(Contacts::First Name;2) returns the contents of the First Name field for record 2 in the Contacts table.

GetNthRecord(Contacts::Has Repetitions[2];2) returns the contents of the second repetition of the Has Repetitions field for record 2 in the Contacts table.

If

Format

If(test;result1;result2)

Parameters

test - any numeric value or logical expression

result1 - expression or field name

result2 - expression or field name

Data type returned
text, number, date, time, timestamp, container
Description

Returns one of two possible results depending on the value of `test`. If `test` is True (any non-zero numeric result), FileMaker Pro returns `result1`. If `test` is False (0), `result2` is returned. `Test` must be an expression that returns either a numeric or Boolean (True, False) result.

Notes

- If you have more than two possible results, consider using the `Case` function.
- By default, if `test` refers to a field that doesn’t yet contain a value, the `If` function returns an empty result. To override this functionality, deselect the Do not evaluate if all referenced fields are empty checkbox.

Examples

If(Country = “USA”; “US Tech Support”; “International Tech Support”) returns International Tech Support, if the Country field contains France or Japan. Returns US Tech Support if the Country field contains USA.

If(State =“CA”; Subtotal * CA Tax Rate; 0) returns the tax if the purchaser is a resident of California, otherwise returns 0.

IsEmpty

Format

IsEmpty(field)

Parameter

field - any field name, text expression, or numeric expression

Data type returned

number

Description

Returns True(1) if field is empty, if a related field, related table, relationship, or file is missing, or if some other error occurs; otherwise returns False(0).

Examples

IsEmpty(OrderNum) returns 1 if the OrderNum field is empty.

If(IsEmpty(LastName); “Invalid record”; ““) displays Invalid Record if the LastName field is blank, but displays nothing if there is an entry in LastName.

IsEmpty(Payments::DatePaid) returns 1 if, for example, the Payments table has been moved or renamed.

IsEmpty(“text”) returns 0.
IsValid

**Format**
IsValid(field)

**Parameter**
field - any field name

**Data type returned**
number

**Description**
Returns False (0) when:

- A record contains an invalid value because of a data type mismatch (text in a date field, for example)
- FileMaker Pro cannot locate (temporarily or permanently) the related table in which the referenced field is defined
- A field has been deleted from a related table, and therefore the references to that field in the parent table are invalid

Otherwise it returns 1 (the data is valid).

**Examples**
IsValid(Datefield) returns 0 if there is non-date data in Datefield, for example if text was imported into it.
IsValid(Amount) returns 0 if there is only text in the number field Amount.
IsValid(table::field) returns 0 if the related table was renamed and the relationship isn’t updated with the new filename.

IsValidExpression

**Format**
IsValidExpression(expression)

**Parameter**
expression - any calculation expression

**Data type returned**
number
**Description**

Returns True (1) if expression syntax is correct. Returns False (0) if expression has a syntax error.

**Examples**

IsValidExpression(calculationField) returns 1 (true) when calculationField contains total + 1.

IsValidExpression(calculationField) returns 0 (false) when calculationField contains abs(-1 with no closing parenthesis.

**Let**

**Format**

Let({[}var1=expression1{;var2=expression2...]});calculation)

**Parameters**

var - any variable name, local variable name, or global variable name (see FileMaker Pro help for guidelines on naming variables)

expression - any calculation expression, field, or constant

calculation - any calculation expression, field, or constant

Parameters in curly braces {} are optional.

**Data type returned**

text, number, date, time, timestamp, container

**Description**

Sets varX to the result of expressionX for the duration of calculation, until the script exits (local variables), or until the file is closed (global variables). Multiple variables are allowed when using a list syntax that is enclosed in square brackets [] and is separated by semicolons. For example:

Let([variable=value;variable2=value2];calculation)

The $ symbol references a local variable and two $$ symbols reference a global variable. An optional repetition number appears in square brackets [] immediately after the variable name. For example:

Let([$variable[repetition]=value;$variable2=value2]{;calculation})
The Let function sets the variables from left to right. You can use previously defined variables (for example, variables that you defined with the Set Variable script step) to define new variable values, and you can nest one Let function within another. If you use a previously defined variable within a nested Let function, the variable has scope only within the nested function (as if you had defined a completely unique variable). See the City example shown below.

Once defined, local and global variables can be referenced in any calculation within their scope. The scope of global variables is limited to the current file. The scope of local variables is the current script. Local variables defined in a calculation are scoped to the file but are only available when scripts are not running. A local and global variable (or even two local variables in different scripts) can have the same name but they are treated as different variables and store different values.

**Examples**

Let (x=5; x*x) returns 25.

Let ([x=5; squared=x*x; cubed=squared*x]; cubed) returns 125.

Let (City=“Paris”; Let (City=“San Francisco”; City&“-“) & City) returns San Francisco - Paris.

The following example sets a local variable counter at repetition 50 with a value of 120: Let ($counter[50]=120; $counter[50]*2) returns 240.

The following example shows how to pass named parameters using the Evaluate, Let, and Get(ScriptParameter) functions, allowing access only to variable 'a' (the example returns 6):

ScriptParameter = "a = 5; b = 10"
Evaluate("Let([" & Get(ScriptParameter) & "]; a+1 )"")

The following example shows how to pass named parameters, allowing access to both variable 'a' and variable 'b'. The simplified first parameter makes the second parameter more complex (the example returns 6, 12):

ScriptParameter = "a = 5; b = 10"
Evaluate("Let([" & Get(ScriptParameter) & "]; a+1 & ", " & b+2 )"")

The following example shows how to pass named parameters, while keeping the ability to check the syntax of the second parameter of the Let function (the example returns 6, 12):

ScriptParameter = "a = 5; b = 10"
Let ([a = Evaluate("Let([" & Get(ScriptParameter) & "]; a )"); b = Evaluate("Let([" & Get(ScriptParameter) & "]; b )"); a+1 & ", " & b+2 )]
Lookup

Format
Lookup(sourceField{;failExpression})

Parameters
sourceField - the field from which the lookup value is taken.
failExpression - any expression.
Parameters in curly braces {} are optional.

Data type returned
text, number, date, time, timestamp, container

Description
Returns the contents of sourceField, using the relationships in the relationships graph. The result of the optional failExpression will be returned if the lookup fails. In order for this function to access the contents of the source field, the tables containing the source field and calculation field need to be related. Calculations using the Lookup function won’t be forced to be unstored calculations.

Example
There are two tables, People and Company, in a database file containing the data shown below.

People table

<table>
<thead>
<tr>
<th>CompanyID</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>John Smith</td>
</tr>
<tr>
<td>200</td>
<td>Peter Wong</td>
</tr>
<tr>
<td>300</td>
<td>Sally Anderson</td>
</tr>
</tbody>
</table>

Company table

<table>
<thead>
<tr>
<th>CompanyID</th>
<th>CompanyName</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Apple</td>
<td>91234</td>
</tr>
<tr>
<td>100</td>
<td>Apple</td>
<td>82345</td>
</tr>
<tr>
<td>200</td>
<td>FileMaker</td>
<td>95054</td>
</tr>
</tbody>
</table>

The People and Company tables are related using the number field CompanyID. The calculation CompanyName = Lookup(Company::CompanyName;“Not found”) defined in the People table will return Apple for the first record, FileMaker for the second record and Not found for the third record.
LookupNext

Format
LookupNext(sourceField;lower/higherFlag)

Parameters
sourceField - the field from which the lookup value is taken.
lower/higherFlag - the keywords lower or higher denote whether the value from the next lower/higher matching record must be taken if the lookup fails.

Data type returned
text, number, date, time, timestamp, container

Description
Returns the value specified in sourceField using the relationships in the relationships graph. If the lookup fails, the value from the source field in the next lower or higher matching record will be returned, as specified by lower/higherFlag. In order for this function to access the value in sourceField, the tables containing the source field and calculation field need to be related. Calculations using the LookupNext function won’t be forced to be unstored calculations.

Example
There are two tables, People and Company, in a database file containing data as shown below.

People table

<table>
<thead>
<tr>
<th>CompanyID</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>John Smith</td>
</tr>
<tr>
<td>200</td>
<td>Peter Wong</td>
</tr>
<tr>
<td>300</td>
<td>Sally Anderson</td>
</tr>
<tr>
<td>377</td>
<td>Mary MacKenzie</td>
</tr>
</tbody>
</table>

Company table

<table>
<thead>
<tr>
<th>CompanyID</th>
<th>CompanyName</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Apple</td>
<td>91234</td>
</tr>
<tr>
<td>100</td>
<td>Apple</td>
<td>82345</td>
</tr>
<tr>
<td>200</td>
<td>FileMaker</td>
<td>95054</td>
</tr>
<tr>
<td>300</td>
<td>Motorola</td>
<td>93456</td>
</tr>
<tr>
<td>400</td>
<td>Cisco</td>
<td>88123</td>
</tr>
</tbody>
</table>
The People and Company tables are related using the number field CompanyID. The calculation \( \text{CompanyName} = \text{LookupNext(Company::CompanyName;Higher)} \) defined in the People table will return Apple, FileMaker, Motorola and Cisco for records 1 to 4.
Number functions are used to manipulate numeric data.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs, page 144</td>
<td>The absolute value (a positive number) of a number.</td>
</tr>
<tr>
<td>Ceiling, page 144</td>
<td>A number rounded up to the next integer.</td>
</tr>
<tr>
<td>Combination, page 145</td>
<td>The number of ways to uniquely choose a specified number of items from a set of a specified size.</td>
</tr>
<tr>
<td>Div, page 145</td>
<td>An integer of the specified number divided by the divisor.</td>
</tr>
<tr>
<td>Exp, page 146</td>
<td>The value of the constant ( e ) (the base of the natural logarithm, equal to 2.7182818) raised to the power of a specified number.</td>
</tr>
<tr>
<td>Factorial, page 146</td>
<td>The factorial of a specified number stopping at 1, or at a specified number factorial.</td>
</tr>
<tr>
<td>Floor, page 147</td>
<td>A number rounded down to the next lower integer.</td>
</tr>
<tr>
<td>Int, page 147</td>
<td>The whole number (integer) part of the value you specify, without rounding.</td>
</tr>
<tr>
<td>Lg, page 148</td>
<td>The base 2 logarithm of the specified number, which can be any positive value.</td>
</tr>
<tr>
<td>Ln, page 149</td>
<td>The base-e (natural) logarithm of the specified number.</td>
</tr>
<tr>
<td>Log, page 149</td>
<td>The common logarithm (base 10) of the specified number, which can be any positive value.</td>
</tr>
<tr>
<td>Mod, page 150</td>
<td>The remainder after a specified number is divided by divisor.</td>
</tr>
<tr>
<td>Random, page 150</td>
<td>A random number between zero and one.</td>
</tr>
<tr>
<td>Round, page 151</td>
<td>A number rounded off to the specified precision (number of decimal places).</td>
</tr>
<tr>
<td>SetPrecision, page 151</td>
<td>Any math functions contained within the specified expression to the specified digits of precision, if the math function supports extended precision.</td>
</tr>
<tr>
<td>Sign, page 152</td>
<td>One of three possible values: -1 when the specified number is negative, 0 when it's zero, and 1 when it's positive.</td>
</tr>
<tr>
<td>Sqrt, page 153</td>
<td>The square root of a number.</td>
</tr>
<tr>
<td>Truncate, page 153</td>
<td>A number truncated to the specified precision (number of decimal places), without evaluating the value of the discarded digits.</td>
</tr>
</tbody>
</table>
Abs

Format
Abs(number)

Parameter
number - any numeric expression or field containing a numeric expression

Data type returned
number, time

Description
Returns the absolute value (a positive number) of number. For example, if a negative number appears in a field, the Abs function removes the minus sign and changes it to a positive value.

Examples
Abs(-123) returns 123.
Abs(PriceDifference) returns the positive value of the number in the PriceDifference field.
Abs(TargetDate - ActualDate) returns a positive value for the number of days difference between the values in TargetDate and ActualDate.

Ceiling

Format
Ceiling(number)

Parameter
number - any numeric expression or field containing a numeric expression

Data type returned
number

Description
Returns number rounded up to the next integer.

Examples
Ceiling(1.25) returns 2.
Ceiling(-1.25) returns -1.
Combination

Format
Combination(setSize;numberOfChoices)

Parameters
setSize - any numeric expression or field containing a non-negative numeric expression.

numberOfChoices - any numeric expression or field containing a non-negative numeric expression.

Data type returned
number

Description
Returns the number of ways to uniquely choose numberOfChoices items from a set of size setSize. The values returned by this function are referred to as combination coefficients. They form Pascal’s triangle. This function is useful in statistics, combinatorics, and polynomial expansions.

\[
\text{Combination} = \frac{\text{Factorial}(\text{setSize}, \text{numberOfChoices})}{\text{Factorial(}\text{numberOfChoices})}
\]

Example
Combination(5;2) returns 10 for a set consisting of \(\{a, b, c, d, e\}\) because the unique choices when choosing two at a time are \(\{ab, ac, ad, ae, bc, bd, be, cd, ce, de\}\).

\((13 \times 12 \times \text{Combination}(4;2) \times \text{Combination}(4;3)) / \text{Combination}(52;5)\) returns 0.00144057..., which is the probability of being dealt a full-house in 5-card poker (less than a 1% chance).

Div

Format
Div(number;divisor)

Parameters
number - any numeric expression or field containing a numeric expression

divisor - any numeric expression or field containing a numeric expression

Data type returned
number
**Description**

Returns the next lowest integer value after dividing `number` by `divisor`. The `Div` function is equivalent to `Floor(number/divisor)`.

**Examples**

`Div(2.5;2)` returns `1`.
`Div(-2.5;2)` returns `-2`.

**Exp**

**Format**

`Exp(number)`

**Parameter**

`number` - any numeric expression or field containing a numeric expression

**Data type returned**

`number`

**Description**

Returns the value of the constant `e` (the base of the natural logarithm, equal to 2.7182818) raised to the power of `number`. The `Exp` function is the inverse of the `Ln` function.

**Examples**

`Exp(1)` returns `2.71828182...`.
`Exp(Ln(2))` returns `2`.
`Exp(0)` returns `1`.

**Factorial**

**Format**

`Factorial(number{;numberOfFactors})`

**Parameters**

`number` - numeric expression or field containing a positive integer.

`numberOfFactors` - any numeric expression or field containing a number that represents how many factors to include in the multiplication.

Parameters in curly braces `{ }` are optional.
Data type returned

number

Description

Returns the factorial of number stopping at 1, or stopping at the optional
numberOfFactors. Useful in statistics and combinatorics.

Where \( n = \text{number} \) and \( i = \text{numberOfFactors} \):

\[
\text{Factorial}(n) = n(n-1)(n-2)\ldots(1)
\]

\[
\text{Factorial}(n;i) = n(n-1)(n-2)\ldots(n-i+1)
\]

Examples

Factorial(3) returns 6, which = 3 * 2 * 1.
Factorial(10;3) returns 720, which = 10 * 9 * 8.

Floor

Format

Floor(number)

Parameter

number - any numeric expression or field containing a numeric expression

Data type returned

number

Description

Returns number rounded down to the next lower integer.

Examples

Floor(1.25) returns 1.
Floor(-1.25) returns -2.

Int

Format

Int(number)
Parameter

number - any numeric expression or field containing a numeric expression

Data type returned

number

Description

Returns the whole number (integer) part of number without rounding. It drops any digits to the right of the decimal point, depending on the number you specify.

Examples

Int(1.45) returns 1.
Int(123.987) returns 123.
Int(Players/3) returns 4, if Players contains 13.

Lg

Format

Lg(number)

Parameter

number - any numeric expression or field containing a numeric expression

Data type returned

number

Description

Returns the base 2 logarithm of number, which can be any positive value. Negative values return an error. For 0, the Lg function returns nothing because these values are out of the acceptable range.

\[ Lg = \frac{\ln(number)}{\ln(2)} \]

Examples

Lg(1) = 0
Lg(2) = 1
Lg(32) = 5
\textbf{Ln}

**Format**
\[ \text{Ln}(\text{number}) \]

**Parameter**
\text{number} - any numeric \textit{expression} or \textit{field} containing a numeric expression

**Data type returned**
\text{number}

**Description**
Returns the base-e (natural) logarithm of \text{number}. The \text{Exp} function is the inverse of the \text{Ln function}. Negative values return an error. For 0, the \text{Ln} function returns nothing because these values are out of the acceptable range.

**Examples**
\text{Ln}(2.7182818) \textit{returns} .99999998...  
\text{Ln}(\text{Exp}(5)) \textit{returns} 5.

\textbf{Log}

**Format**
\[ \text{Log}(\text{number}) \]

**Parameter**
\text{number} - any positive numeric \textit{expression} or \textit{field} containing a numeric expression

**Data type returned**
\text{number}

**Description**
Returns the common logarithm (base 10) of \text{number}, which can be any positive value. Negative values return an error. For 0, the \text{Log} function returns nothing because these values are out of the acceptable range.

\[ \text{Log} = \frac{\text{Ln}(\text{number})}{\text{Ln}(10)} \]

**Examples**
\text{Log}(1) \textit{returns} 0.  
\text{Log}(100) \textit{returns} 2.
Mod

Format
Mod(number;divisor)

Parameters
number - any numeric expression or field containing a numeric expression
divisor - numeric expression or field containing a numeric expression

Data type returned
number

Description
Returns the remainder after number is divided by divisor.
Use the Mod function to test whether a number is even or odd by specifying a divisor of 2 (if the result is zero the number is even, otherwise it's odd).

\[
\text{Mod} = \text{number} - (\text{Div(number;divisor)} \times \text{divisor})
\]

Examples
Mod(13;4) returns 1.
Mod(7;5) returns 2.
Mod(7;-5) returns -3.
Mod(-7;5) returns 3.
Mod(-7;-5) returns -2.
Mod(Participants;TeamSize) returns 4 if Participants contains 40 and TeamSize contains 9.
If(Mod(Get(RecordNumber);2) = 0;"even";"odd") labels a record even or odd using the Get(RecordNumber) function.

Random

Format
Random

Parameter
None

Data type returned
number
**Description**
Returns a random number between zero and one. FileMaker Pro generates a new random number when you:
- insert the `Random` function into a formula
- cause a formula containing the `Random` function to be reevaluated (by changing data in any of the fields the formula uses)
- display or access a calculation field defined to have an unstored result.

**Example**
```
NumDice + NumSides * Random
```

**Round**

**Format**
```
Round(number;precision)
```

**Parameters**
- `number` - any numeric expression or field containing a numeric expression
- `precision` - any numeric expression or field containing a numeric expression

**Data type returned**
```
number
```

**Description**
Returns number rounded off to the specified precision (number of decimal places). If you round a negative number of decimal places, all digits to the right of the decimal point are dropped, and the number is rounded to the nearest tens, hundreds, and so on. The `Round` function always rounds up at 0.5.

**Examples**
- `Round(123.456;2)` returns 123.46.
- `Round(14.5;0)` returns 15.
- `Round(29343.98;-3)` returns 29000.
- `Round(123.456;-1)` returns 120.

**SetPrecision**

**Format**
```
SetPrecision(expression;precision)
```
Parameters
expression - any numeric expression
precision - any number or numeric expression

Data type returned
number

Description
Computes any math functions contained within expression that support extended precision to precision decimal places (up to 400). All functions except the trigonometric functions support extended precision. This function doesn’t perform a truncation: constant numbers are left at the precision in which they were entered.

Examples
SetPrecision(5/9;30) returns 0.555555555555555555555555555556.
SetPrecision(If(field1>5;Exp(50);Average(5/9;1/7;5/7));25) returns either
5184705528587072464087.4533229334853848274691006 if field1 > 5, or
0.4708994708994708994708995 if field1 <= 5.

Sign

Format
Sign(number)

Parameter
number - any numeric expression or field containing a numeric expression

Data type returned
number

Description
Returns one of three possible values: -1 when number is negative, 0 when it’s zero, and 1 when it’s positive.

Examples
Sign(15.12) returns 1.
Sign(-175) returns -1.
Sign(BalanceDue) returns 0, if BalanceDue is a number field containing 0.
**Sqrt**

**Format**

\[ \text{Sqrt}(\text{number}) \]

**Parameter**

- `number`: any positive number, numeric expression, or field containing a numeric expression.

**Data type returned**

- `number`

**Description**

Calculates the square root of `number`.

\[ \text{Sqrt} = \sqrt{\text{number}} \]

**Examples**

- `\text{Sqrt}(4)` returns 2.
- `\text{Sqrt}(\text{SquareFeet})` returns 6 if the `SquareFeet` number field contains 36.

**Truncate**

**Format**

\[ \text{Truncate}(\text{number};\text{precision}) \]

**Parameters**

- `number`: any numeric expression or field containing a numeric expression
- `precision`: any numeric expression or field containing a numeric expression

**Data type returned**

- `number`

**Description**

Returns `number` truncated to the specified `precision` (number of decimal places), without evaluating the value of discarded digits. Use the `Round` function to round up or down to the required precision.

**Examples**

- `\text{Truncate}(123.456;2)` returns 123.45.
- `\text{Truncate}(-14.6;0)` returns -14.
Truncate (29343.98; -3) returns 29000.
Truncate (123.456; 4) returns 123.456.
Truncate (29343.98; 5) returns 29343.98.
Chapter 10
Repeating functions

Repeating functions perform calculations on repeating fields.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extend, page 156</td>
<td>In a calculation involving both repeating and non-repeating fields, allows a value in a non-repeating field to be used with every repetition in a repeating field.</td>
</tr>
<tr>
<td>GetRepetition, page 156</td>
<td>The contents of the specified repetition of a repeating field.</td>
</tr>
<tr>
<td>Last, page 157</td>
<td>The last valid, non-blank value in the specified field.</td>
</tr>
</tbody>
</table>
Extend

Format
Extend(non-repeatingField)

Parameter
non-repeatingField - any non-repeating field (a field defined to contain only one value), or an expression that returns a reference to one.

Data type returned
text, number, date, time, timestamp, container

Description
Allows a value in non-repeatingField to be used with every repetition in a repeating field. Use the Extend function with calculations involving both repeating and non-repeating fields. Without the Extend function, the value in non-repeatingField is used only with the first repetition in the repeating field.

Examples
Extend(TaxRate) * Quantity * ItemPrice returns 1.197, .6606, and 1.497 when TaxRate contains .06; the repeating field Quantity contains 1, 3, and 5; and the repeating field ItemPrice contains 19.95, 3.67, and 4.99.

Item Count * Extend(if(Company Size > 100; Reduced Price; Price)) returns $1250, $500, and $750 when Reduced Price contains $50; the repeating field Item Count contains 25, 10, and 15; and Company Size is greater than 100. If Company Size is less than 100 and Price contains $100, this calculation returns $2500, $1000, and $1500.

GetRepetition

Format
GetRepetition(repeatingField; number)

Parameters
repeatingField - any repeating field, or an expression that returns a reference to a repeating field.
number - the field repetition number.

Data type returned
text, number, date, time, timestamp, container
Description
Returns the contents of the repeating field specified by number.

Examples
ParcelBids is a field defined to repeat with ten values and contains the values 2500, 1200, and 1500.
GetRepetition(ParcelBids;2) returns 1200.
GetRepetition(if(IsEmpty(ParcelBids) ≠ true, ParcelBids, HouseBids);2) returns 1200.
GetRepetition(ParcelBids;5) returns nothing.
Note You can also find the contents of a particular repetition in a repeating field using square brackets [] as array operators. For example, ParcelBids[2] returns 1200. See FileMaker Pro help.

Last

Format
Last(field)

Parameter
field - any repeating field or related field, or an expression that returns a reference to a repeating field or related field.

Data type returned
text, number, date, time, timestamp, container

Description
Returns the last valid, non-blank value in field. If field specifies a repeating field then it returns the last non-blank repetition. If field specifies a related field, then it returns the last non-blank value in the related set.

Note The last related value will depend on the way related records are sorted. If the related records are not sorted, then the Last function returns a value based on the creation order of the records.

Examples
Last(ParcelBids) returns 1500 if ParcelBids is a number field defined to repeat with ten values and contains the values 2500, 1200, and 1500.
Last(Payments::PaymentDate) returns the payment date in the last matching record in the Payments table.
Last(if(IsEmpty(Company);PersonalPhone;WorkPhone)) returns the last non-empty phone number from the repeating field PersonalPhone when the Company field is empty. If the Company field is not empty, the function returns the last non-empty phone number from the repeating field WorkPhone.
Chapter 11  
Summary functions

Summary functions produce a summary of all records in the found set, or subsummary values for records in different groups. Formulas can contain more than one summary function. Summary functions calculate more slowly than other functions because they generate values for a range of records.

An alternate way to generate similar calculated results is to use Aggregate functions to summarize data in related records (whether or not they appear in a portal). See chapter 2, “Aggregate functions” and information about summarizing data in portals.

Click the function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetSummary, page 160</td>
<td>The value of the summary field for the current range of records when the database file is sorted by the break field.</td>
</tr>
</tbody>
</table>
GetSummary

**Format**
GetSummary(summaryField; breakField)

**Parameters**
- `summaryField` - field of type `summary`, or an expression that returns a reference to one.
- `breakField` - field, or an expression that returns a reference to one. To calculate a grand summary value, use the same summary field for both the summary field and the `breakField` parameters.

The GetSummary function must be set up in the same table as the break field.

**Data type returned**
number, date, time, timestamp

**Description**
Returns the value of `summaryField` for the current range of records when the database file is sorted by `breakField`. This produces subsummary values. If the database file isn’t sorted by the break field, the result is blank.

When a summary field is also used as the break field, returns the summary field value for the entire found set of records (a grand summary value).

Use the GetSummary function to capture summary values when you want to:
- Use summary values in a calculation
- Display subsummary values in Browse mode or in a body part

Calculations using the GetSummary function are unstored.

**Note** You can get similar results using a self-join relationship and “Aggregate functions” on page 17. For more information, see FileMaker Pro help.

**Examples**
GetSummary(Total Sales; Country) returns a summary of all records pertaining to the value in the Country field.

GetSummary(Total Sales, if(Number of Countries > 1, Country, Sales Zone)) returns a summary of Total Sales by Country if Number of Countries is greater than 1. Otherwise, it returns a summary of Total Sales by Sales Zone.

GetSummary(Total Sales; Total Sales) produces a summary of all records (similar to using a summary field, which is a total of total sales).

If(ThisCharge > 3 * GetSummary(AvgCharge; Customer), “Verify this charge”, “”) displays Verify this charge if the current charge is greater than three times the average charge.
Chapter 12  
Text functions

Text functions can be used to analyze, rearrange, extract, and build text strings. For example, you could use the MiddleWords function to extract specific words from supplied text.

Text functions operate on these parameters:

- fields of type text
- text constants (in quotes)
- expressions having a text result

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact</strong>, page 163</td>
<td>1 (True) for an exact match, or 0 (False) for a mismatch between two text strings or container fields.</td>
</tr>
<tr>
<td><strong>Filter</strong>, page 164</td>
<td>Only the specified characters, in the order that they were originally entered in the text.</td>
</tr>
<tr>
<td><strong>FilterValues</strong>, page 164</td>
<td>Only the specified values, in the order that they were originally entered in the text.</td>
</tr>
<tr>
<td><strong>GetAsCSS</strong>, page 165</td>
<td>The specified text, converted to the CSS (Cascading Style Sheets) format.</td>
</tr>
<tr>
<td><strong>GetAsDate</strong>, page 166</td>
<td>Dates in the specified text as field type date, for use in formulas involving dates or date functions.</td>
</tr>
<tr>
<td><strong>GetAsNumber</strong>, page 166</td>
<td>Numbers in the specified text as field type number, for use with formulas involving numbers or numeric functions.</td>
</tr>
<tr>
<td><strong>GetAsSVG</strong>, page 167</td>
<td>The specified text, converted to the SVG (Scalable Vector Graphics) format.</td>
</tr>
<tr>
<td><strong>GetAsText</strong>, page 168</td>
<td>The specified number, date, time or timestamp as field type text, for use with formulas involving text or text functions.</td>
</tr>
<tr>
<td><strong>GetAsTime</strong>, page 168</td>
<td>Times or timestamps in the specified text as field type time, for use with formulas involving the time or timestamp functions.</td>
</tr>
<tr>
<td><strong>GetAsTimestamp</strong>, page 169</td>
<td>The specified data as field type timestamp, for use with formulas involving timestamps.</td>
</tr>
<tr>
<td><strong>GetAsURLEncoded</strong>, page 169</td>
<td>The specified text, converted with URL (Uniform Resource Locators) encoding.</td>
</tr>
<tr>
<td><strong>GetValue</strong>, page 170</td>
<td>A specific value from a list of values.</td>
</tr>
<tr>
<td><strong>Hiragana</strong>, page 171</td>
<td>Hiragana converted from Katakana (hankaku and zenkaku).</td>
</tr>
<tr>
<td><strong>KanaHankaku</strong>, page 171</td>
<td>Hankaku Katakana converted from Zenkaku Katakana.</td>
</tr>
<tr>
<td><strong>KanaZenkaku</strong>, page 172</td>
<td>Zenkaku Katakana converted from Hankaku Katakana.</td>
</tr>
<tr>
<td>This function</td>
<td>Returns</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>KanjiNumeral, page 172</td>
<td>Kanji numerals converted from Arabic numerals.</td>
</tr>
<tr>
<td>Katakana, page 172</td>
<td>Zenkaku Katakana converted from Hiragana.</td>
</tr>
<tr>
<td>Left, page 173</td>
<td>The specified number of characters in the text, counting from the left.</td>
</tr>
<tr>
<td>LeftValues, page 173</td>
<td>The specified number of values in the text, counting from the left.</td>
</tr>
<tr>
<td>LeftWords, page 174</td>
<td>The specified number of words in the text, counting from the left.</td>
</tr>
<tr>
<td>Length, page 175</td>
<td>The number of characters in the specified text, including all spaces, numbers, and special characters.</td>
</tr>
<tr>
<td>Lower, page 175</td>
<td>All letters in the specified text as lowercase.</td>
</tr>
<tr>
<td>Middle, page 176</td>
<td>The specified number of characters in the text, starting at a specified character position.</td>
</tr>
<tr>
<td>MiddleValues, page 176</td>
<td>The specified number of values in the text, starting with a specified value.</td>
</tr>
<tr>
<td>MiddleWords, page 177</td>
<td>The specified number of words in the text, starting with a specified word.</td>
</tr>
<tr>
<td>NumToJText, page 178</td>
<td>Roman numbers converted from Japanese text.</td>
</tr>
<tr>
<td>PatternCount, page 179</td>
<td>The number of occurrences of a search string in the specified text.</td>
</tr>
<tr>
<td>Position, page 179</td>
<td>The specified occurrence of a search string, starting from a specified position.</td>
</tr>
<tr>
<td>Proper, page 180</td>
<td>The first letter of each word in the specified text as uppercase, and all other letters as lowercase.</td>
</tr>
<tr>
<td>Quote, page 181</td>
<td>The specified text surrounded by quotation marks (“ “).</td>
</tr>
<tr>
<td>Replace, page 181</td>
<td>A new string of characters consisting of the specified text as modified by the specified replacement text.</td>
</tr>
<tr>
<td>Right, page 182</td>
<td>The specified number of characters in the text, counting from the right.</td>
</tr>
<tr>
<td>RightValues, page 182</td>
<td>The specified number of values in the text, counting from the right.</td>
</tr>
<tr>
<td>RightWords, page 183</td>
<td>The specified number of words in the text, counting from the right.</td>
</tr>
<tr>
<td>RomanHankaku, page 184</td>
<td>Hankaku (alphanumeric &amp; symbols) converted from Zenkaku (alphanumeric &amp; symbols).</td>
</tr>
<tr>
<td>RomanZenkaku, page 184</td>
<td>Zenkaku (alphanumeric &amp; symbols) converted from Hankaku (alphanumeric &amp; symbols).</td>
</tr>
<tr>
<td>SerialIncrement, page 185</td>
<td>The combined text and numbers in a specified value, with the numbers incremented by the specified amount.</td>
</tr>
<tr>
<td>Substitute, page 186</td>
<td>A text string with every occurrence of a specified search string in the text replaced by a specified replacement string.</td>
</tr>
<tr>
<td>Trim, page 186</td>
<td>Text stripped of all leading and trailing spaces.</td>
</tr>
<tr>
<td>TrimAll, page 187</td>
<td>Text with full width spaces between non-Roman and Roman characters removed.</td>
</tr>
<tr>
<td>Upper, page 188</td>
<td>All letters in the specified text as uppercase.</td>
</tr>
</tbody>
</table>
This function | Returns
--- | ---
ValueCount, page 189 | A count of the total number of values in the specified text.
WordCount, page 189 | A count of the total number of words in the specified text.

**Exact**

**Format**

Exact(originalText;comparisonText)

**Parameters**

originalText - any text expression, text field, or container field

comparisonText - any text expression, text field, or container field

**Data type returned**

number

**Description**

Compares the contents of any two fields. For text to match exactly, the uppercase and lowercase usage must be the same. If the fields match, the result is 1 (True); otherwise the result is 0 (False). For container fields, the data must be stored in the same manner (either embedded, or stored by file reference).

**Note** When evaluating values, text attributes such as font, styles, and sizes are not considered.

**Tip** If case isn't important, use the Lower or Upper function on both parameters to process data before checking for an exact match.

**Examples**

Exact(“McDonald”;“McDonald”) returns 1 (True).

Exact(“McDonald”;“MCDONALD”) returns 0 (False).

Exact(Upper(“McDonald”);Upper(“MCDONALD”)) returns 1 (True).

Exact(“John”;“John”) returns 0 (False).

Exact(BillTo;ShipTo) returns 1 (True) when the value in BillTo is the same as the value in ShipTo.

Exact(Recipient;Upper(Recipient)) returns 1 (True), when Recipient contains “JOHNSON”

Exact(Country;“Spain”) returns 1 (True) when the Country field contains Spain.
Filter

Format
Filter(textToFilter; filterText)

Parameters

- **textToFilter**: any text expression or text field
- **filterText**: the characters to preserve in the specified text

Data type returned
text

Description
Returns from `textToFilter` only those characters specified in `filterText`, in the order that they were originally entered in `textToFilter`. If `filterText` doesn’t have any characters, an empty string is returned. The `Filter` function is case-sensitive.

Examples

Filter(“(408) 555-1212”; “0123456789”) returns 4085551212.
Filter(“AaBb”; “AB”) returns AB.

The following example removes all text from the provided data, then formats the remaining numbers in the preferred phone number formatting:

Let(phone = filter(theField; “0123456789”); “(“ & left(phone; 3) & “)’” & middle(phone; 4; 3) & “-” & middle(phone; 7; 4))

If `theField` contains Work: 408.555.1212 this calculation returns (408)555-1212.

FilterValues

Format
FilterValues(textToFilter; filterValues)

Parameters

- **textToFilter**: any text expression or text field
- **filterValues**: values that you want to preserve in the specified text

Important See chapter 4, “Design functions” for information about literal text parameters.

Data type returned
text
Description

Returns a text result containing only the characters that were provided in filterValues, in the order that they were originally entered in textToFilter. If filterValues doesn’t have any characters, an empty string is returned.

Values are text items separated by carriage returns. A value can be empty, a single character, a word, a sentence, or a paragraph. When you press Enter or Return, you start creating a new value. The last value will be recognized with or without a carriage return.

When the textToFilter or the filterValues parameter is a literal string, you must insert a paragraph character (¶) between each item in the string. To insert a carriage return character, click the ¶ button in the Specify Calculation dialog box.

Examples

FilterValues(“Plaid¶Canvas¶Suitcase”;“Plaid¶Canvas”) returns

Plaid
Canvas

FilterValues(ValueListItems(“Database”;“Sizes”);“Medium¶Small”) returns

Small
Medium

when a database file named Database has a value list Sizes that contains Small¶Medium¶Large.

GetAsCSS

Format

GetAsCSS(text)

Parameter

text - any text expression or text field

Data type returned

text

Description

Returns text, converted to the CSS (Cascading Style Sheets) format, an Internet text format similar to HTML. CSS supports more text formats than HTML, so CSS will represent what you have typed more accurately.
**Example**

GetAsCSS(text) returns the example result shown below when the field text contains the word “Frank” and the word Frank has the following text attributes: Font = Helvetica, Font Size = 12 points, Font Color = red, Font Style = bold.

Example result:

<SPAN STYLE = “font-family: ‘Helvetica’;font-size: 12px;color: #FF0000;font-weight: bold;text-align: left;”>Frank</SPAN>

**GetAsDate**

**Format**

GetAsDate(text)

**Parameter**

text - any text expression or text field containing text in the same format as the date on the system where the file was created.

**Data type returned**

date

**Description**

Returns dates in text as data type date, for use in formulas involving dates or date functions. The format of text date must be the same as the date format on the system where the file was created.

Use the GetAsDate or Date function to enter a date constant into a formula.

**Note** If the function returns a number instead of a date, go to the Specify Calculation dialog box and make sure the Calculation result is date.

**Important** To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see FileMaker Pro help.

**Example**

GetAsDate(“03/03/2005”) returns 3/3/2005. You can perform date operations on this result.

**GetAsNumber**

**Format**

GetAsNumber(text)

**Parameter**

text - any text expression or text field containing numbers
Data type returned

number

Description

Returns only the numbers in text, as data type number, for use with formulas involving numbers or numeric functions. The GetAsNumber function drops all non-numeric characters from text.

Examples

GetAsNumber(“FY98”) returns 98.
GetAsNumber(“$1,254.50”) returns 1254.5.
GetAsNumber(“2 + 2”) returns 22.
GetAsNumber(SerialNumber) returns 35684, when the value of SerialNumber is TKV35FRG6HH84.

GetAsSVG

Format

GetAsSVG(text)

Parameter

text - any text expression or text field

Data type returned

text

Description

Returns text, converted to the SVG (Scalable Vector Graphics) format, an Internet text format similar to HTML or CSS. SVG supports more text formats than HTML, so SVG will represent what you have typed more accurately.

Example

GetAsSVG(text) returns the example result (below) when the field text contains the word “Frank” and the word Frank has the following text attributes: Font = Helvetica, Font Size = 12 points, Font Color = red, Font Style = bold.

Example result:

<StyleList>
  <Style#0>"font-family: ‘Helvetica’;font-size: 12px;color: #FF0000;font-weight: bold;text-align: left;”,
  Begin: 1, End: 5</Style>
GetAsText

Format
GetAsText(data)

Parameter
data - any number, date, time or timestamp expression, or a field containing a number, date, time, timestamp, or container.

Data type returned
text

Description
Returns data as data type text, for use with formulas involving text or text functions. data can be data type number, date, time, timestamp, or container.

For a container field, GetAsText returns external path information, text (when the container contains text that does not resolve into a valid path), or a question mark (?) if the container data is embedded in the database.

Examples
GetAsText(45) returns 45.
“You are “ & GetAsText(DaysDelinquent) & “ days late.” returns You are 3 days late. when the value of DaysDelinquent is 3.
“FY” & GetAsText(FiscalYear) returns FY98, if the FiscalYear number field contains 98.

GetAsTime

Format
GetAsTime(text)

Parameter
text - any text expression or text field containing a time

Data type returned
time
Description
Returns times or timestamps in text as data type time, for use with formulas involving the Time or Timestamp functions. The format of the supplied time must be the same as the time format on the system where the file was created.

Use the GetAsTime or the Time function to enter a time constant into a formula.

Examples
GetAsTime(“02:47:35”) returns 2:47:35 when you select time as the calculation result. You can perform time calculations on this result.

GetAsTime(“02:47:35”) returns 1/1/0001 2:47:35 when you select timestamp as the calculation result.

Abs(GetAsTime(“12:15 pm”) - CheckOut) returns 3:00:00 when the CheckOut time field contains 3:15 PM.

GetAsTimestamp

Format
GetAsTimestamp(text)

Parameter
text - any text expression, or text, number, date, or time field

Data type returned
timestamp

Description
Returns text as data type timestamp, for use with formulas involving timestamps. Text strings must be in the form of a date followed by a time. A number is considered to be the number of seconds since 1/1/0001. There are 86400 seconds in each day.

Examples
GetAsTimestamp(50000) returns 1/1/0001 1:53:20 PM.

GetAsURLEncoded

Format
GetAsURLEncoded(text)

Parameter
text - any text expression or text field
**Data type returned**

text

**Description**

Returns text as URL (Uniform Resource Locator) encoding, for use as a URL. Removes all styles from text.

All characters are first converted to UTF-8 format. Characters that are neither letters nor digits, or digits that are in the upper ASCII range, are converted to %HH format (a percent sign followed by the character’s hexadecimal value).

See the following website for more information on URL encoding:

http://www.w3.org

**Examples**

GetAsURLEncoded("Hello") returns Hello
GetAsURLEncoded("San Francisco") returns San%20Francisco
GetAsURLEncoded("français") returns fran%c3%a7ais

**GetValue**

**Format**

GetValue(listOfValues;valueNumber)

**Parameter**

listOfValues - a list of carriage return-delimited values

valueNumber - the value to return from the list

**Data type returned**

text

**Description**

Returns the requested value given by valueNumber from listOfValues. Useful in looping scripts or recursive custom calculations.

Values are text items separated by carriage returns. You can place several values together to create a carriage return-delimited list of values. A value can be empty, a single character, a word, a sentence, or a paragraph. When you press Enter or Return, you start creating a new value. The last value will be recognized with or without a carriage return.

When the listOfValues parameter is a literal string, you must insert a literal carriage return character (¶) between each item in the string. To insert a literal carriage return character, click the ¶ button in the Specify Calculation dialog box.
Example
GetValue("London¶Paris¶Hong Kong");2) returns Paris

Hiragana

Format
Hiragana(text)

Parameter
text - any text expression or text field

Data type returned
text

Description
Converts Katakana (hankaku and zenkaku) in text to Hiragana.

Example
Hiragana("アイウエオ") returns あいうえお

KanaHankaku

Format
KanaHankaku(text)

Parameter
text - any text expression or text field

Data type returned
text

Description
Converts Zenkaku Katakana to Hankaku Katakana.

Example
KanaHankaku("データベース") returns データベース
KanaZenkaku

**Format**
KanaZenkaku(text)

**Parameter**
text - any text expression or text field

**Data type returned**
text

**Description**
Converts Hankaku Katakana to Zenkaku Katakana.

**Example**
KanaZenkaku("\"ﾃﾞ ﾍ ﾄ\"") returns データベース

KanjiNumeral

**Format**
KanjiNumeral(text)

**Parameter**
text - any text expression or text field

**Data type returned**
text

**Description**
Converts Arabic numerals to Kanji numeral.

**Examples**
KanjiNumeral(123) returns 一二三
KanjiNumeral("\"富士見台２の３の２5\"") returns 富士見台二の三の二十五

Katakana

**Format**
Katakana(text)
Parameter
text - any text expression or text field

Data type returned
text

Description
Converts from Hiragana to Zenkaku Katakana.

Example
Katakana(“あいうえお”) returns アイウエオ

Left

Format
Left(text;numberOfCharacters)

Parameters
text - any text expression or text field
numberOfCharacters - any numeric expression or field containing a number

Data type returned
text

Description
Returns the numberOfCharacters in text, counting from the left.

Examples
Left(“Manufacturing”;4) returns Manu.
Left(Name;Position(Name;“;1;1)) returns Sophie, when the Name field contains Sophie Tang.
Left(PostalCode;3) & Upper(Left(LastName;4)) returns 481JOHN when the PostalCode field contains 48187 and LastName contains Johnson.

LeftValues

Format
LeftValues(text;numberOfValues)

Parameters
text - any text expression or text field
**numberOfValues** - any numeric expression or field containing a number

**Important** See chapter 4, “Design functions” for information about literal text parameters.

**Data type returned**
text

**Description**
Returns a text result containing the specified `numberOfValues` from the list of values in `text`, counting from the left.

Values are text items separated by carriage returns. A value can be empty, a single character, a word, a sentence, or a paragraph. When you press Return you start creating a new value. The last value will be recognized with or without a carriage return.

Each returned value ends with a carriage return, allowing lists to be easily concatenated.

**Examples**

```
LeftValues("Plaid¶Canvas¶Suitcase";2) returns
Plaid
Canvas
```

```
LeftValues(list;1) returns
Sophie
when the text being evaluated contains
Sophie
Bill
```

**LeftWords**

**Format**

`LeftWords(text;numberOfWords)`

**Parameters**

- `text` - any text expression or text field
- `numberOfWords` - any numeric expression or field containing a number

**Data type returned**
text

**Description**
Returns a text result containing the `numberOfWords` in `text`, counting from the left.

**Note** The ampersand (&) and hyphen (-) characters identify the beginning of a new word.
Examples
LeftWords(“Plaid Canvas Suitcase”;2) returns Plaid Canvas.
LeftWords(Name;1) returns Sophie, when the Name field contains Sophie Tang.

Length

Format
Length(field)

Parameter
field - any text, number, date, time, timestamp, or container field, or any text expression or numeric expression

Data type returned
number

Description
Returns the number of characters in field, including all spaces, numbers, and special characters. For a container field, Length returns the total stored size of objects in bytes.

Examples
Length(“John”) returns 4.
Length(Description) returns 12 when the value in Description is Modem for PC.
Length(“M1” & Left(Product;5)) returns 7, when the Product field contains Canvas Backpack.

Lower

Format
Lower(text)

Parameter
text - any text expression or text field

Data type returned
text

Description
Returns all letters in text as lowercase.
Examples

Lower(“ABCD”) returns abcd.
Lower(Course) returns history, when the Course field contains History.
Lower(“YOUR BILL IS OVERDUE”) returns your bill is overdue.

Middle

Format

Middle(text; start; numberOfCharacters)

Parameters

text - any text expression or text field
start - any numeric expression or field containing a number
numberOfCharacters - any numeric expression or field containing a number

Data type returned
text

Description

Extracts the numberOfCharacters from text, starting at the character position specified by start.

Examples

Middle(“(408)555-9054”;2;3) returns 408.
Middle(PhoneNumber;2;3) returns 408 when the PhoneNumber field contains (408) 555-9054.
Middle(“abcdefgij”;5;2) returns ef.
Middle(Name;Position(Name;“ “;1;1)+1;3) returns Smi, when the text field Name contains John Smith.

MiddleValues

Format

MiddleValues(text;startingValue;numberOfValues)

Parameters

text - any text expression or text field
startingValue - any numeric expression or field containing a number
numberOfValues - any numeric expression or field containing a number
Important See chapter 4, “Design functions” for information about literal text parameters.

Data type returned

text

Description

Returns a text result containing the specified `numberOfValues in text`, starting with `startingValue`.

Values are text items separated by carriage returns. A value can be empty, a single character, a word, a sentence or a paragraph. When you press Return you start creating a new value. The last value will be recognized with or without a carriage return.

Each value that is returned ends with a carriage return, allowing lists to be easily concatenated.

Examples

MiddleValues(“Plaid¶Canvas¶Suitcase”;2;1) returns Canvas

MiddleValues(list;2;2) returns

Bill

John

when the list field contains

Sophie

Bill

John

MiddleWords

Format

MiddleWords(text;startingWord;numberOfWords)

Parameters

text - any text expression or text field

startingWord - any numeric expression or field containing a number

numberOfWords - any numeric expression or field containing a number

Data type returned

text
Description
Returns a text result containing the `numberOfWords` from `text`, beginning at `startingWord`.

**Note** The ampersand (`&`) and hyphen (`-`) characters identify the beginning of a new word.

Examples
MiddleWords(“Plaid Canvas Suitcase”;2;2) returns Canvas Suitcase.
MiddleWords(Name;1;2) returns Brigitte Erika, when the Name field contains Brigitte Erika Durand.

**NumToJText**

**Format**
`NumToJText(number;separator;characterType)`

**Parameters**
- `number` - any numeric **expression** or field containing a number
- `separator` - a number from 0 - 3 representing a separator
- `characterType` - a number from 0 - 3 representing a type

**Data type returned**
text

**Description**
Converts Roman numbers in `number` to Japanese text. If the value for `separator` and `characterType` are blank or other than 0 to 3, then 0 is used.

**Separator:**
- 0 - no separator
- 1 - every 3 digits (thousands)
- 2 - ten thousands (万) and millions (億) unit
- 3 - tens (十), hundreds (百), thousands (千), ten thousands (万) and millions (億) unit

**Type:**
- 0 - half width (Hankaku) number
- 1 - full width (Zenkaku) number
- 2 - Kanji character number ——三
- 3 - Traditional-old-style Kanji character number 亁弐參
Examples

NumToJText(123456789; 2; 0) returns 1 億 2345 万 6789
NumToJText(123456789; 3; 2) returns 1 億 二千三百四十五万六千七百八十九

PatternCount

Format

PatternCount(text; searchString)

Parameters

text - any text expression or text field
searchString - any text expression or text field representing the set of characters you want to find

Data type returned

number

Description

Returns the number of occurrences of searchString in text.

Examples

PatternCount(“Mississippi”; “is”) returns 2.
PatternCount(“Mississippi”; “issi”) returns 1 (the function isn’t inclusive).
PatternCount(Attending; “Guest”) returns 1 if the Guest checkbox is one of the items selected in the Attending field.

Position

Format

Position(text; searchString; start; occurrence)

Parameters

text - any text expression or text field
searchString - any text expression or text field representing the set of characters you want to find.
start - any numeric expression, or field containing a number, representing the number of characters from the start of the text string at which to begin the search.
occurrence - any numeric expression or field containing a number, representing which instance of the text string you want to find. A negative occurrence value causes the scan to go in the opposite direction from start. A zero value for occurrence is invalid and returns a result of zero.

**Data type returned**

number

**Description**

Returns the starting position of the specified occurrence of `searchString` in `text`. If `searchString` isn't contained in `text` or if there was no specified occurrence, zero is returned. The `Position` function is not case-sensitive.

**Examples**

Position(“Mississippi”;“iss”;1;1) returns 2.
Position(“Mississippi”;“iss”;1;2) returns 5.
Position(“Mississippi”;“iss”;3;1) returns 5.

Left(Name;Position(Name;“ “;1;1)-1) returns William, when Name is a text field that contains William Smith.
Right(Name;Length(Name) - Position(Name;“ “;Length(Name);-1)) returns Smith.

**Proper**

**Format**

Proper(text)

**Parameter**

text - any text expression or text field

**Data type returned**

text

**Description**

Returns the first letter of each word in `text` as uppercase and all other letters as lowercase.

**Examples**

Proper(“ABCD”) returns Abcd.
Proper(Name) returns Yumiko Kitagawa, when the Name field contains YUMIKO KITAGAWA.
Quote

Format
Quote(text)

Parameter
text - any text expression or field

Data type returned
text

Description
Returns the text form of text enclosed in quotation marks. Special characters within text are escaped appropriately. This function protects text from being evaluated by the Evaluate function.

Examples
Quote(“hello”) returns “hello”
Quote(“abc¶”) returns “abc¶”
Quote(“say \"hello\" fred”) returns “say \"hello\" fred”
Evaluate(Quote(“1 + 2”)) returns 1 + 2
Evaluate(“1 + 2&” & Quote(“ - 1 + 2”)) returns 3 - 1 + 2

Replace

Format
Replace(text;start;numberOfCharacters;replacementText)

Parameters
text - any text expression or text field
start - any numeric expression or field containing a number representing the starting position in text.
numberOfCharacters - any numeric expression or field containing a number representing the number of characters to remove from text.
replacementText - any text expression or field containing the text to replace in the original string.

Data type returned
text
Description
Replaces a string of characters in text with replacementText. Character replacement in text begins at the start character position and continues for numberOfCharacters characters. Compare to the Substitute function.

Examples
Replace(“1234567”;5;1;“X”) returns 1234X67.
Replace(“1234567”;5;1;“XX”) returns 1234XX67.
Replace(“1234567”;5;2;“X”) returns 1234X7.
Replace(“William”;3;4;“NEW TEXT”) returns WiNEW TEXTm.
Replace(PhoneNumber;1;3;“415”) returns 415-555-9054, when the PhoneNumber field contains 408-555-9054.

Right

Format
Right(text;numberOfCharacters)

Parameters
text - any text expression or text field
numberOfCharacters - any numeric expression or field containing a number

Data type returned
text

Description
Returns the specified numberOfCharacters in text, counting from the right.

Examples
Right(“Manufacturing”;4) returns ring.
Right(Name;Length(Name) - Position(Name;“ “;1;1)) returns Cannon, when the Name field contains Michelle Cannon.
Right(SerialNumber;3) & Upper(Left(LastName;4)) returns 187FERR when the SerialNumber text field contains 00-48-187 and LastName contains Ferrini.

RightValues

Format
RightValues(text;numberOfValues)
Parameters

text - any text expression or text field

numberOfValues - any numeric expression or field containing a number

Important See chapter 4, “Design functions” for information about literal text parameters.

Data type returned
text

Description

Returns a text result containing the specified numberOfValues in text starting from the right.

Values are text items separated by carriage returns. You can place several items together to create a carriage return-delimited list of values. A value can be empty, a single character, a word, a sentence, or a paragraph. When you press Return you start creating a new value. The last value will be recognized with or without a carriage return.

When the text parameter is a literal string as in the example below, you must insert a literal carriage return character between each item in the list. In the Specify Calculation dialog box, click the ¶ button to insert a literal carriage return character.

Each value that is returned ends with a carriage return, allowing lists to be easily concatenated.

Examples

RightValues(“Plaid¶Canvas¶Suitcase”;2) returns

Canvas

Suitcase

RightValues(names;1) returns

John

when the names field contains

Sophie

Bill

John

RightWords

Format

RightWords(text;numberOfWords)

Parameters

text - any text expression or text field
numberOfWords - any numeric expression or field containing a number

Data type returned
text

Description
Returns a text result containing the numberOfWords in text, counting from the right.

Note The ampersand (&) and hyphen (-) characters identify the beginning of a new word.

Examples
RightWords(“Plaid Canvas Suitcase”;2) returns Canvas Suitcase.
RightWords(Name;1) returns Virtanen, when the Name field contains Matti Virtanen.

RomanHankaku

Format
RomanHankaku(text)

Parameter
text - any text expression or text field

Data type returned
text

Description
Converts from Zenkaku alphanumeric and symbols to Hankaku alphanumeric and symbols.

Example
RomanHankaku(“M a c i n t o s h”) returns Macintosh

RomanZenkaku

Format
RomanZenkaku(text)

Parameter
text - any text expression or text field

Data type returned
text
Description
Converts from Hankaku alphanumeric and symbols to Zenkaku alphanumeric and symbols.

Examples
RomanZenkaku(“Macintosh”) returns Macintosh

SerialIncrement

Format
SerialIncrement(text;incrementBy)

Parameters
text - any text that also contains a number
incrementBy - any numeric expression to increment the text by

Data type returned
text

Description
Returns the combined text and numbers specified by text, with the numbers in text incremented by the specified amount.

The text in text isn’t removed, as normally happens when performing standard math against a value that contains text.

If the incrementBy value is a decimal number, then only the integer portion of incrementBy value is added to the last number in text. Any character other than a number is considered a separator. You can use both positive and negative incrementBy values.

Examples
SerialIncrement(“abc12”;1) returns abc13.
SerialIncrement(“abc12”;7) returns abc19.
SerialIncrement(“abc12”;-1) returns abc11.
SerialIncrement(“abc12”;1.2) returns abc13.
SerialIncrement(“abc1.2”;1.2) returns abc1.3.

In the example below any character other than a number is considered as a separator and the number on the far right is incremented.
SerialIncrement(“abc123;999”;1) returns abc123;1000.
Substitute

Format
Substitute(text;searchString;replaceString)

Parameters
- text - any text expression or text field
- searchString - any text expression or text field
- replaceString - any text expression or text field

Data type returned
text

Description
Returns a text string with every occurrence of searchString in text replaced by replaceString in text.

The Substitute function is case-sensitive. Compare to the Replace function.

Multiple substitutions are allowed when you enclose each pair of searchString and replaceString parameters within square brackets [ ] and separate them with semicolons. Each search and replace list item is also separated by semicolons. For example:

Substitute(text; [search1; replace1]; [search2; replace2]; ... [searchN; replaceN])

Examples
Substitute(Description;“WYSIWYG.”;“What you see is what you get”) replaces every occurrence of the acronym “WYSIWYG.” in the Description field with the phrase What you see is what you get.

Substitute(text;[“a”;“A”];[“b”;“B”]) replaces every lowercase a or b with A or B.

Trim

Format
Trim(text)

Parameter
text - any text expression or text field

Data type returned
text
Description

Returns text stripped of all leading and trailing spaces.

Tip Use the Trim function to remove unneeded spaces when you convert files from other programs or systems that require a fixed number of characters per field, or to remove spaces accidentally typed during data entry.

Examples

Trim(“ Tom ”) returns Tom.

Trim(Middle(“00230013 William 1234”;9;9)) returns William.

TrimAll

Format

TrimAll(text;trimSpaces;trimType)

Parameters

- text - any text expression or text field
- trimSpaces - 0 or False, 1 or True
- trimType - 0 through 3 depending on the trim style that you wish to use

Data type returned

text

Description

Returns a copy of text with all leading and trailing spaces removed.

Set trimSpaces to True (1) if you want to include the removal of full-width spaces between non-Roman and Roman characters. Set trimSpaces to False (0) if you do not.

A character is considered Roman if its unicode value is less than U+2F00. Any character whose unicode value is greater than or equal to U+2F00 is considered non-Roman.

Characters within the Roman range are those belonging to the following character blocks: Latin, Latin-1 Supplement, Latin Extended-A & B, IPA Extensions, Spacing Modifier Letters, Combining Diacritical Marks, Greek, Cyrillic, Armenian, Hebrew, Arabic, Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayalam, Thai, Lao, Tibetan, Georgian, Hangul Jamo, and additional Latin and Greek extended blocks.

Symbols within the Roman range include punctuation characters, superscripts, subscripts, currency symbols, combining marks for symbols, letter-like symbols, number forms, arrows, math operators, control pictures, geometric shapes, dingbats, and so on.
Characters within the non-Roman range are those belonging to the CJK symbols/punctuations area, Hiragana, Katakana, Bopomofo, Hangul compatibility Jamo, Kanbun, CJK unified ideographs, and so on.

Spaces are removed or inserted depending on the value of \texttt{trimType}, as given in the following tables:

<table>
<thead>
<tr>
<th>This \texttt{trimType} value</th>
<th>Does this</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Removes all spaces between non-Roman and Roman characters (always leave one space between Roman words).</td>
</tr>
<tr>
<td>1</td>
<td>Always includes a half-width space between non-Roman and Roman characters (always leave one space between Roman words).</td>
</tr>
<tr>
<td>2</td>
<td>Removes spaces between non-Roman characters (reduce multiple space between non-Roman and Roman words to 1 space; do not add spaces if there are none; always leave one space between Roman words).</td>
</tr>
<tr>
<td>3</td>
<td>Removes all spaces everywhere.</td>
</tr>
</tbody>
</table>

In all cases, spaces between non-Roman characters are removed.

<table>
<thead>
<tr>
<th>Type</th>
<th>Non-Roman - Non-Roman</th>
<th>Non-Roman - Roman</th>
<th>Roman - Roman</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Remove</td>
<td>Remove</td>
<td>1 space</td>
</tr>
<tr>
<td>1</td>
<td>Remove</td>
<td>1 space*</td>
<td>1 space</td>
</tr>
<tr>
<td>2</td>
<td>Remove</td>
<td>1 space</td>
<td>1 space</td>
</tr>
<tr>
<td>3</td>
<td>Remove</td>
<td>Remove</td>
<td>Remove</td>
</tr>
</tbody>
</table>

* = insert space between non-Roman and Roman text if there isn’t one.

**Examples**

\texttt{TrimAll(名前,1,0)} returns 山田 太郎 if the value of 名前 field is 山田 太郎

\texttt{TrimAll(ファイルメーカーPro は高品质*,1,0)} returns ファイルメーカーProは高品质

**Upper**

**Format**

\texttt{Upper(text)}

**Parameter**

text - any \texttt{text expression} or text \texttt{field}

**Data type returned**

text
Description
Returns all letters in text as uppercase. Use the Upper function to ensure consistent data entry of such things as state abbreviations or postal codes.

Examples
Upper(“Ca”) returns CA.
Upper(“12n34p”) returns 12N34P.

ValueCount

Format
ValueCount(text)

Parameter
text - any text expression or text field

Important See chapter 4, “Design functions” for information about literal text parameters.

Data type returned
number

Description
Returns a count of the total number of values in text.

Values are text items separated by carriage returns. You can place several items together to create a carriage-return-delimited list of values. A value can be empty, a single character, a word, a sentence, or a paragraph. When you press Return you start creating a new value. The last value will be recognized with or without a carriage return.

When the text parameter is a literal string as in the example below, you must insert a literal carriage return character between each item in the list. In the Specify Calculation dialog box, click the ¶ button to insert a literal carriage return character.

Examples
ValueCount(“Item 1¶Item 2¶Item 3”) returns 3.
ValueCount(ValueListItems(“Employees”;“Employee Names”) returns the total number of values in the Employee Names value list in the Employees database file.

WordCount

Format
WordCount(text)
Parameter
text - any text expression or text field

Data type returned
number

Description
Returns a count of the total number of words in text.

Examples
WordCount(“The sun is rising.”) returns 4.
WordCount(Letter) returns the total number of words in the Letter field.

Note The ampersand (&) and hyphen (-) characters identify the beginning of a new word.
Chapter 13
Text formatting functions

Text formatting functions can be used to change the color, font, size, and style of the specified text. For example, you could use the TextFont function to change the font of the specified text from Arial to Courier.

Text formatting functions operate on these parameters:

- fields of type text
- text constants (in quotes)
- expressions having a text result

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB, page 192</td>
<td>A number obtained by combining the red, green, and blue values to represent a color.</td>
</tr>
<tr>
<td>TextColor, page 192</td>
<td>The color of text to the color specified by the RGB function.</td>
</tr>
<tr>
<td>TextColorRemove, page 193</td>
<td>Text with the font colors reverted to the default font color for the field.</td>
</tr>
<tr>
<td>TextFont, page 194</td>
<td>Text in the specified font and character set.</td>
</tr>
<tr>
<td>TextFontRemove, page 195</td>
<td>Text with the fonts reverted to the default font for the field.</td>
</tr>
<tr>
<td>TextFormatRemove, page 196</td>
<td>Text with the formatting reverted to the default text format for the field.</td>
</tr>
<tr>
<td>TextSize, page 197</td>
<td>Text in the specified font size.</td>
</tr>
<tr>
<td>TextSizeRemove, page 197</td>
<td>Text with the font sizes reverted to the default font size for the field.</td>
</tr>
<tr>
<td>TextStyleAdd, page 198</td>
<td>Text with the specified styles added in a single action.</td>
</tr>
<tr>
<td>TextStyleRemove, page 199</td>
<td>Text with the specified styles removed in a single action.</td>
</tr>
</tbody>
</table>
RGB

Format
RGB(red;green;blue)

Parameters
- red - any numeric expression or numeric field containing a value ranging from 0 to 255
- green - any numeric expression or numeric field containing a value ranging from 0 to 255
- blue - any numeric expression or numeric field containing a value ranging from 0 to 255

Data type returned
number

Description
Returns an integer number from 0 to 16777215 obtained by combining the red, green, and blue values (each ranging from 0 to 255) to represent a color. Numbers returned by this function can be passed as the color parameter in the TextColor or TextColorRemove functions. The RGB function uses the following formula to calculate the result:
\[ \text{red} \times 256^2 + \text{green} \times 256 + \text{blue} \]
where \(256^2 = 65536\)

Examples
- RGB(255;0;0) returns 16711680 representing red.
- RGB(0;255;0) returns 65280 representing green.
- RGB(0;0;255) returns 255 representing blue.
- RGB(0;0;0) returns 0 representing black.
- RGB(255;255;255) returns 16777215 representing white.

TextColor

Format
TextColor(text;RGB(red;green;blue))

Parameters
- text - any text expression or text field
- RGB(red;green;blue) - any integer number from 0 to 16777215 obtained by combining the red, green, and blue values (each ranging from 0 to 255) to represent a color
Data type returned

text

Description

Changes the color of text to the color specified by the RGB function.

**Note** Text formatting options will be lost if the data type that is returned is something other than text.

Examples

TextColor("Plaid";RGB(255;0;0)) returns the word Plaid in red.
TextColor("Plaid";RGB(0;255;0)) returns the word Plaid in green.
TextColor("Plaid";RGB(0;0;255)) returns the word Plaid in blue.
TextColor("Plaid";RGB(0;0;0)) returns the word Plaid in black.

TextColorRemove

Format

TextColorRemove(text{;RGB(red;green;blue)})

Parameters

text - any text expression or text field

RGB(red;green;blue) - any integer number from 0 to 16777215 obtained by combining the red, green, and blue values (each ranging from 0 to 255) to represent a color

Parameters in curly braces {} are optional.

Data type returned

text

Description

Removes all font colors in text, or removes the font color specified by the RGB function. If you don’t specify a color, all of the text displays in the default font color that was set in Layout mode for the field. When the font color is specified by the RGB function, only the specified font color is removed from every portion of the text displayed in that color and these same portions of the text are then displayed in the field's default font color.

**Note** Text formatting options will be lost if the data type that is returned is something other than text.
Examples
TextColorRemove(“Red Text and Green Text”) returns Red Text and Green Text displayed in the field’s default font color.
TextColorRemove(“Red Text and Green Text”;RGB(255;0;0)) returns Red Text and Green Text with only the pure red font color removed from the words Red Text.

TextFont

Format
TextFont(text;fontName{;fontScript})

Parameters
text - any text expression or text field.
fontName - any font name expressed in text.
{fontScript} - the name of a character set that contains characters required for writing in the specified language.

Parameters in curly braces {} are optional.

Note The fontScript parameter is not enclosed in quotation marks (“ “), and can have any of the values listed below in Description.

Data type returned
text

Description
Changes the font of text to the specified fontName or optional {fontScript}. Spellings for font names must be correct and are case-sensitive. Text formatting options will be lost if the data type that is returned is something other than text.

FileMaker Pro looks for a font that matches the specified font name and font script character set. If no matches exist, FileMaker Pro looks for a default font with the font script specified in the Fonts tab of the Preferences dialog box. If this fails, then the TextFont function uses the default font for the system script specified in the Fonts tab of the Preferences dialog box. This font might not be the same as the font script provided.

The following font scripts are available:
• Roman
• Greek
• Cyrillic
• CentralEurope
• ShiftJIS
• TraditionalChinese
• SimplifiedChinese
• OEM
• Symbol
• Other

Examples
TextFont(“Plaid”;“Courier”) returns the word Plaid in the Courier font.
TextFont(“Plaid”;“Arial”) returns the word Plaid in the Arial font.
TextFont(“Plaid”;“Arial”;Cyrillic) returns the word Plaid in the Arial font in the font script of Cyrillic.

TextFontRemove

Format
TextFontRemove(text{;fontName;fontScript})

Parameters
text - any text expression or text field
fontName - any font name expressed in text
fontScript - the name of a character set that contains characters required for writing in the specified language

Parameters in curly braces {} are optional.

Note The fontScript parameter is not enclosed in quotation marks (“ “), and can have any of the values listed below in Description.

Data type returned
text

Description
Removes all fonts in text, or removes the font specified by fontName or the combination of fontName and fontScript. If you don't specify a font, all of the text displays in the default font that was set in Layout mode for the field. When the font is specified by fontName or the combination of fontName and fontScript, only the specified font is removed from every portion of the text displayed in that font and these same portions of the text are then displayed in the field's default font.

Spellings for font names must be correct and are case-sensitive. Text formatting options will be lost if the data type that is returned is something other than text.
FileMaker Pro looks for a font that matches the specified font name and font script character set. If no matches exist, FileMaker Pro looks for a default font with the font script specified in the Fonts tab of the Preferences dialog box. If this fails, then the `TextFontRemove` function uses the default font for the system script specified in the Fonts tab of the Preferences dialog box. This font might not be the same as the font script provided.

The following font scripts are available:

- Roman
- Greek
- Cyrillic
- CentralEurope
- ShiftJIS
- TraditionalChinese
- SimplifiedChinese
- OEM
- Symbol
- Other

**Examples**

`TextFontRemove(“Arial Text and Courier Text”)` returns Arial Text and Courier Text displayed in the field’s default font.

`TextFontRemove(“Arial Text and Courier Text”;“Arial”)` returns Arial Text and Courier Text with the Arial font removed from the words Arial Text for all fontScripts that use the Arial font.

`TextFontRemove(“Arial Text and Courier Text”;“Arial”;Cyrillic)` returns Arial Text and Courier Text with the Arial font removed from Cyrillic character sets.

**TextFormatRemove**

**Format**

`TextFormatRemove(text)`

**Parameters**

- `text` - any text expression or text field

**Data type returned**

text
Description
Removes all text formatting from text in a single action. All fonts, styles, font sizes, and font colors are removed from the specified text.

Example
TextFormatRemove(“Plaid”) returns the word Plaid without any text formatting applied.

TextSize

Format
TextSize(text;fontSize)

Parameters
text - any text expression or text field
fontSize - any font size expressed as an integer

Data type returned
text

Description
Changes the font size of the specified text to fontSize. The font size is described in points (72 points to the inch). Text formatting options will be lost if the data type that is returned is something other than text.

Examples
TextSize(“Plaid”;18) returns the word Plaid in 18 point text.
TextSize(“Plaid”;24) returns the word Plaid in 24 point text.

TextSizeRemove

Format
TextSizeRemove(text{;sizeToRemove})

Parameters
text - any text expression or text field
sizeToRemove - any font size expressed as an integer

Parameters in curly braces {} are optional.

Data type returned
text
Description

Removes all font sizes in `text`, or removes the font size specified by `sizeToRemove`. If you don’t specify a size, all of the text displays in the default font size that was set in Layout mode for the field. When the font size is specified by `sizeToRemove`, only the specified font size is removed from every portion of the text displayed in that size and these same portions of the text are then displayed in the field's default font size.

The font size is described in points (72 points to the inch). Text formatting options will be lost if the data type that is returned is something other than text.

Examples

```plaintext
TextSizeRemove("10 Point Text and 18 Point Text") returns 10 Point Text and 18 Point Text displayed in the field’s default font size.

TextSizeRemove("10 Point Text and 18 Point Text";18) returns 10 Point Text and 18 Point Text with the 18 point font size removed from the words 18 Point Text.
```

TextStyleAdd

Format

`TextStyleAdd(text;styles)`

Parameters

- `text` - any text expression or text field
- `styles` - any named style listed below in Description

Data type returned

text

Description

Adds the specified `styles` to `text` in a single action. You can add multiple styles by using the `+` operator between style names. Negative values are not valid. All styles will be removed, if the only style specified is Plain. Plain is ignored if mixed with other styles. Styles are not case-sensitive and do not contain spaces.

Text formatting options will be lost if the data type that is returned is something other than text.

The styles that are available are:

- Plain
- Bold
- Italic
- Underline
- Condense
• Extend
• Strikethrough
• SmallCaps
• Superscript
• Subscript
• Uppercase
• Lowercase
• Titlecase
• WordUnderline
• DoubleUnderline
• AllStyles (all available styles)

Examples

TextStyleAdd("Plaid";Italic) returns the word Plaid in italics.

TextStyleAdd(FirstName;Bold+Underline) returns Sophie in bold, underlined text when the FirstName field contains Sophie.

The following calculation removes all styles from the text, then italicizes the entire phrase.

TextStyleAdd(TextStyleAdd(FirstName;Plain);Italic)

The following calculation creates two descriptions of styles, then concatenates two phrases using these styles. Using the Let function is an effective way to avoid creating a long and complex TextStyleAdd statement.

Let([TitleStyle=Smallcaps+Titlecase;BodyStyle=Plain]; TextStyleAdd(titleField;titleStyle) &"¶¶" & TextStyleAdd(bodyField;BodyStyle))

In the following example, you might want to find every occurrence of several words and change their style. Using the Substitute function combined with the TextStyleAdd function is a good way to accomplish this goal.

Substitute(ArticleBody;["Phrase1";TextStyleAdd("Phrase 1";Italic)];["Phrase 2";TextStyleAdd("Phrase 2";Bold)];)

TextStyleRemove

Format

TextStyleRemove(text;styles)

Parameters

- text - any text expression or text field
- styles - any named style from the list of available styles
Data type returned
text

Description
Removes the specified styles from text in a single action. You can remove multiple styles by using the + operator between style names. Negative values are not valid. The Plain styles cannot be used for this function. Plain is ignored if intermingled with other styles. Styles are not case-sensitive and do not contain spaces.

An additional style called AllStyles has been provided to make it easier to remove all styles. Text formatting options will be lost if the data type that is returned is something other than text.

The styles that are available are:

- Plain
- Bold
- Italic
- Underline
- Condense
- Extend
- Strikethrough
- SmallCaps
- Superscript
- Subscript
- Uppercase
- Lowercase
- Titlecase
- WordUnderline
- DoubleUnderline
- AllStyles (all available styles)

Examples
TextStyleRemove("Plaid";Italic) returns the word Plaid with the italics style removed.

TextStyleRemove(FirstName;Bold + Underline) returns Sophie with the bold and underlined styles removed when the FirstName field contains Sophie.

TextStyleRemove(FirstName;AllStyles) returns Sophie without any styles.
Chapter 14
Time functions

Time functions calculate times and manipulate time information.
Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hour, page 202</td>
<td>A number representing the number of hours in a time value.</td>
</tr>
<tr>
<td>Minute, page 202</td>
<td>A number representing the number of minutes in a time value.</td>
</tr>
<tr>
<td>Seconds, page 203</td>
<td>A number representing the number of seconds in a time value.</td>
</tr>
<tr>
<td>Time, page 203</td>
<td>A time result with the specified number of hours, minutes, and seconds.</td>
</tr>
</tbody>
</table>
Hour

Format
Hour(time)

Parameter
time - any time value or field of type time

Data type returned
number

Description
Returns a number representing the number of hours in time.

Examples
Hour("12:15:23") returns 12.
Hour(Duration) + (Minute(Duration)/60) returns 2.5, when the Duration time field contains 2:30:15.
If(Hour(HoursWorked) > 8;"Overtime Pay";"") returns Overtime Pay when the number of hours in HoursWorked is greater than 8.
Hour(CheckIn) returns 3 when the value of CheckIn is 3:24.

Minute

Format
Minute(time)

Parameter
time - any time value or field of type time

Data type returned
number

Description
Returns a number representing the number of minutes in time.

Examples
Minute("12:15:23") returns 15.
Hour(Duration) + (Minute(Duration)/60) returns 2.5, if the Duration time field contains 2:30:15.
Seconds

**Format**
Seconds(time)

**Parameter**
time - any time value or field of type time

**Data type returned**
number

**Description**
Returns a number representing the number of seconds in time.

**Examples**
Hour(Duration) + (Minute(Duration)/60) + (Seconds(Duration)/3600 returns 2.504166, if the Duration time field contains 2:30:15.

Time

**Format**
Time(hours;minutes;seconds)

**Parameters**
hours - the hour value of a time
minutes - the minutes value of a time
seconds - the seconds value of a time

**Data type returned**
time

**Description**
Returns a time result with the specified number of hours, minutes, and seconds. FileMaker Pro compensates when you supply fractional hours or minutes. The result is the time, formatted according to the time format of the field in the current layout.

Use the Time function or the GetAsTime function to enter a time constant into a formula.
Examples
Time(4;14;32) returns 4:14:32.
Time(4.5;10;30) returns 4:40:30.
Time(4;15;70) returns 4:16:10.
Chapter 15

Timestamp functions

Timestamps are used for a wide variety of synchronization purposes, such as marking the exact date and time at which a particular event occurred.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp, page 206</td>
<td>A timestamp containing a calendar date and time of day.</td>
</tr>
</tbody>
</table>
Timestamp

Format
Timestamp(date;time)

Parameters
date - any calendar date or date field
time - any time value or time field

Data type returned
timestamp

Description
Returns a timestamp containing date as a calendar date and time as a time of day. The format of the result depends on the date and time formats that were in use when the database file was created. You can change the date and time formats in the Regional Settings Control Panel (Windows 2000), the Date and Time Control Panel (Windows XP), or the Date & Time System Preference (Mac OS).

Examples
Timestamp(Date(10;11;2004);Time(9;10;30)) returns 10/11/2004 9:10:30AM.
Timestamp(Date(10;11;2004);Time(13;10;30)) returns 10/11/2004 1:10:30PM.
Timestamp(Date(10;11;2004);Time(10;65;5)) returns 10/11/2004 11:05:05AM.
Timestamp(Date(10;35;2004);Time(4;5;6)) returns 11/4/2004 4:05:06AM.
Trigonometric functions are used to calculate degrees, angles, and other geometric data.

**Note** All trigonometric functions use radians as the unit of measure. Once you have a result, you can convert the radians into degrees using the Degrees function.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atan, page 208</td>
<td>The trigonometric arc tangent (inverse tangent) of radians.</td>
</tr>
<tr>
<td>Cos, page 208</td>
<td>The cosine of the specified angle.</td>
</tr>
<tr>
<td>Degrees, page 209</td>
<td>Degrees, converted from the specified radians.</td>
</tr>
<tr>
<td>Pi, page 209</td>
<td>The value of the constant Pi, which is approximately 3.14159.</td>
</tr>
<tr>
<td>Radians, page 210</td>
<td>Radians, converted from the specified degrees.</td>
</tr>
<tr>
<td>Sin, page 210</td>
<td>The sine of the specified angle.</td>
</tr>
<tr>
<td>Tan, page 211</td>
<td>The tangent of the specified angle.</td>
</tr>
</tbody>
</table>
Atan

**Format**
Atan(number)

**Parameter**
number - any numeric expression or field containing a numeric expression

**Data type returned**
number

**Description**
Returns the trigonometric arc tangent (inverse tangent) of number. The arc tangent is the angle, in radians, whose tangent is equal to the specified number.

**Examples**
Atan(1) returns 0.78539816....
Degrees(Atan(1)) returns 45.

Cos

**Format**
Cos(angleInRadians)

**Parameter**
angleInRadians - any numeric expression or field containing a numeric expression, in radians.

**Data type returned**
number

**Description**
Returns the cosine of angleInRadians. The specified angle must be represented in radians.

**Examples**
Cos(1.047) returns 0.50017107....
Cos(Radians(60)) returns 0.5.
Degrees

Format

Degrees(angleInRadians)

Parameter

angleInRadians - any numeric expression or field containing a numeric expression, in radians.

Data type returned

number

Description

Converts angleInRadians to degrees. Use this function to translate results from trigonometric functions from radians to degrees.

\[ \text{Degrees} = \frac{180 \cdot \text{angleInRadians}}{\pi} \]

Examples

Degrees(Atan(1)) returns 45.

Degrees(1.0472) returns 60.00014030....

Pi

Format

Pi

Parameter

None

Data type returned

number

Description

Calculates the value of the constant Pi (π), which is approximately 3.14159.

Example

Pi * 15 returns 47.124.
Radians

Format
Radians(angleInDegrees)

Parameter
angleInDegrees - any numeric expression or field containing a numeric expression, in degrees.

Data type returned
number

Description
Converts angleInDegrees to radians. The parameters for FileMaker Pro trigonometric functions must be expressed in radians. If the values you want to use as parameters in a trigonometric equation are in degrees, use this function to convert them to radians first. A degree is equal to Pi/180 radians.

\[
\text{Radians} = \frac{\pi \cdot \text{angleInDegrees}}{180}
\]

Examples
Radians(45) returns .78539816....
Sin(Radians(30)) returns .5.

Sin

Format
Sin(angleInRadians)

Parameter
angleInRadians - any numeric expression or field containing a numeric expression, in radians.

Data type returned
number

Description
Returns the sine of angleInRadians expressed in radians.

Examples
Sin(Radians(60)) returns .86602.
Sin(.610865) returns .57357624....
Tan

Format
Tan(angleInRadians)

Parameter
angleInRadians - any numeric expression or field containing a numeric expression, in radians.

Data type returned
number

Description
Returns the tangent of angleInRadians.

Note With the Tan function, you cannot use values exactly equal to 90 degrees (Pi/2 radians), or multiples of 90 degrees.

\[
\text{Tan} = \frac{\text{Sin}(\text{angleInRadians})}{\text{Cos}(\text{angleInRadians})}
\]

Examples
Tan(.13) returns .13073731....
Tan(Radians(34)) returns .6745085.
A

Access key (Windows)
A key that activates a menu, menu item, or control when used with the ALT key. In Windows, this key corresponds to the underlined letter on a menu, command, or dialog box option.

Access privileges
Permission to view and work with certain records, fields, layouts, value lists, and scripts and to perform selected activities in a file.

Account
A username and (usually) password that accesses a file with a defined level of privileges. There are two pre-defined accounts: Admin and Guest. Admin is a Full Access account that can be renamed or deleted. At least one Full Access account that is authenticated via FileMaker must be defined for each database file. Guest account is a special account that cannot be renamed or deleted, but can be made active or inactive.

ActiveX Automation
A Windows programming and scripting protocol that allows external control of specific commands and actions in FileMaker Pro, including opening and closing FileMaker Pro databases, toggling the application's visibility, and performing FileMaker Pro scripts.

AppleScript
A scripting language you can use to control functions of the Mac OS and of applications that support AppleScript (often called scriptable applications).

Apple events
A Mac OS technology that lets applications communicate with one another. FileMaker Pro can send and receive Apple events to and from applications that support them.

Ascending sort order
Alphabetical sequence (A to Z) for words, lowest to highest order for numbers, and earliest to latest for dates and times.

ASCII character set
American Standard Code for Information Interchange. A standard character set used by computer systems worldwide (often extended for different alphabets).
Authentication
The process of checking the validity of an account and password (if one is defined) before assigning privileges and allowing access to a system or a database file. An account authenticated via FileMaker Pro or FileMaker Server is referred to as a FileMaker Account. (FileMaker Server can also authenticate an account via External Server -- an external authentication system such as Apple Open Directory, or a Windows Domain.)

B
Blank layout
A predefined layout that contains empty body, header, and footer parts.

Body part
A layout part that contains individual records from a database file.

Book
In the status area, a control for moving from one record to another in Browse mode, from one layout to another in Layout mode, from one find request to another in Find mode, and from one page to another in Preview mode.

If you don't see the status area, click the status area control at the bottom of the document window.

Boolean value
A Boolean value is either True or False. A field containing any number except zero evaluates as True. A field containing zero, no data, or content that does not resolve into a number evaluates as False. For example, a field containing “ABC,” “ABC0,” or an empty field is False. A field containing “1” or “ABC2” is True.

Break field
In a subsummary part, records are grouped (sorted) by values in another field, called the break field. Whenever the value of the break field changes, the report "breaks" and FileMaker Pro inserts the subsummary part.

Browse mode
The FileMaker mode in which you enter and edit information in fields. Groups of fields make up the records of your database. You can either view one record at a time (choose View menu > View as Form), or view your records in a list (choose View menu > View as List), or view records arranged in a spreadsheet-like table (choose View menu > View as Table).

(Use Browse mode to enter and edit your information; use Layout mode to design how your information is displayed. Use Find mode to find records that match search criteria; use Preview mode to display how your records will print.)
Button
Any layout object (a 3-D rectangle with a text label if created by the Button tool) that performs a specified script in Browse or Find modes.

C
Cache
The amount of memory assigned to FileMaker Pro. A larger cache size increases performance. A smaller cache size saves data to the hard disk more frequently, offering greater protection in case of a system crash.

Calculation field
A field that returns the result of a calculation of values. You can create a formula for the calculation using functions, constants, operators, and information from other fields in the same record.

Cascading style sheets
A system of codes or tags that define how a web browser displays information in a web page. Cascading style sheets provide more control over the layout and appearance of web pages than HTML. Cascading style sheets work like templates for web pages. If a web page contains cascading style sheets, users must view it in a browser that supports cascading style sheets.

CGI (Common Gateway Interface)
The specification for communication between an HTTP server and server gateway programs, which is supported by most servers.

Character encoding
The character set or code page of a file. If necessary, you can specify a character set to be used when importing, exporting, indexing, sorting, and spell-checking files. FileMaker supports ASCII, Windows ANSI, Macintosh, Japanese (Shift-JIS), Unicode UTF-8, Unicode UTF-16, and Unicode UTF-16 Windows.

Client
A user that opens a database file that is shared on a network, published in a browser, or shared via ODBC/JDBC. FileMaker Network settings and privileges determine how clients interact with databases hosted through FileMaker Pro, FileMaker Server, and FileMaker Server Advanced.

Client application
The application that requests data (using SQL) from a data source (using ODBC or JDBC). Also, FileMaker Pro is a client application when it accesses a database hosted by FileMaker Server.
Client/server architecture
The relationship between two networked computers that share resources. The client requests services from the server, and the server provides services to the client.

Clipboard
A temporary storage area in computer memory where FileMaker Pro places the most recent selection you've cut or copied.

Clone
A copy of a FileMaker Pro file that contains all the field definitions, tables, layouts, scripts, and page setup options, but none of the data.

Column
When a database file is viewed as a table, a column corresponds to a field.

Columnar List/Report layout
A predefined layout type for setting up simple reports (columnar or extended columnar) or complex reports with grouped data (subsummary reports). The fields that you specify appear in columns across the screen or page in one line. Field names are in the header part and the footer part is blank.

Combo box
A type of drop-down list you can set up in Layout mode. In the Field/Control Setup dialog box, select Include arrow to show and hide list. The list will only drop when users click the arrow, not when they enter the field.

Commit
To save changes to a database file. Certain actions such as navigating between records, finding, and sorting do not change the file's modification date. Other actions, such as changing data in a record or changing a layout do change the file's modification date.

Constant
In a formula, an unchanging value. For example, a constant can be a field name, a text literal ("Total:"), or a number. The value of the constant doesn't change from record to record as the formula is evaluated. Text constants in formulas can be up to 253 characters long.

Container data type
Pictures, sounds, QuickTime movies, OLE objects (Windows), and files of any type can be inserted in a container field.

Context
The starting point or perspective from which calculations and scripts are begun, and from which a relationship is evaluated in the relationships graph.

Convert
Opening a data file from another application, which creates a new FileMaker Pro file containing the data.
Also refers to opening a file created with a previous version of FileMaker Pro.

D

Data Entry Only privilege set
One of the three pre-defined privilege sets that appear in every file. The Data Entry Only privilege set allows read/write access to the records in a file, but not design access (for example, the ability to create layouts and value lists).

Data source
The data you want to access (such as a DBMS) and information to locate the data (such as the path or IP address).

Database Design Report
A FileMaker Pro Advanced tool that creates a report of your database schema.

Database file
A collection of information in a file containing one or more tables pertaining to a subject, such as customers or invoices. (A large database can also comprise many database files.)

Database Management System (DBMS)
An application that allows users to store, process, and retrieve information in a database.

Descending sort order
Reverse alphabetical sequence (Z to A) for words, highest to lowest order for numbers, and latest to earliest dates and times.

Developer Utilities
A FileMaker Pro Advanced feature that lets you bind your files into a runtime solution, display the files in Kiosk mode, customize FileMaker menus, and automatically rename and update links in related files.

Domain name
The primary subdivision of Internet addresses, which is indicated by the last part of an Internet address after the final period (or dot). In the United States, the standard domains are .com, .edu, .gov, .mil, .org, and .net. In other countries, the top-level domain is usually the country domain.

Driver
The ODBC or JDBC driver translates SQL queries into commands that a DBMS can understand. It processes ODBC/JDBC calls, submits SQL requests to the data source, and returns the data back to the driver manager, which then routes it to the requesting application (for example, FileMaker Pro).
**Driver manager**

The control panel that manages communication between requesting applications and data sources. When an application makes a request via ODBC/JDBC, the driver manager routes the request through the correct driver to the correct data source and returns the data to the requesting application. All ODBC/JDBC drivers and data sources to be used on that computer are registered with the driver manager.

**Drop-down calendar**

A field set up to display an interactive monthly calendar when a user enters the field in Browse or Find mode.

**DTD (Document Type Definition)**

A formal description of a particular type of XML. It defines a document structure, including the names of data elements and where they may occur within the structure. Valid XML conforms to the rules established in its DTD. XML parsers (such as Xerces) can check the validity of XML to its DTD.

---

**E**

**Email**

Electronic mail. A system for transmitting messages from one computer or terminal to another. A message sent from one computer user to another is stored in the recipient’s account mailbox until that person logs onto the system and reads the message.

**Embedded OLE object**

An embedded OLE object is part of your database file. It can be text, a graphic object, a sound file, or a movie created in another application. FileMaker Pro launches that other application when you view or work with the object.

**Envelope layout**

A predefined layout with fields arranged for printing on standard business envelopes.

**Ethernet**

A type of fast local area network used for connecting computers and peripherals within the same building or campus.

**EXIF (Exchangeable image file)**

A standardized digital camera format for including additional data with each photo, such as the date and time the photo was created, aperture, shutter speed, and other information about each photo.

**Export**

To save data from one file so that it can be used in another file or in another application.
**Expression**

A value or any computation that produces a value. Expressions can contain functions, field values, and constants and can be combined to produce other expressions.

**Extended privilege**

Data sharing permissions that determine if a privilege set allows users to open a shared file using FileMaker Pro or FileMaker Server (fmapp), view a database as an ODBC or JDBC data source (fmxdbc), or view a database using a web browser via Instant Web Publishing (fmiwp), XML web publishing (fmxml), XSLT web publishing (fmxslt), or FileMaker Mobile (fmmobile). Plug-ins from third-party developers may provide additional extended privileges.

**External script**

A script used by a database file, but defined in a different database file. Use the Perform Script script step to select a defined script from a related file, or to select a file reference to a database file on your hard drive or network.

---

**F**

**Field**

The basic unit of data in a record. You define a field to hold a specific, discrete category of data, such as Last Name, Employee Photo, or to display the result of a calculation. You can define text, number, date, time, timestamp, container, calculation, and summary fields. Field can also refer to the object on a layout that displays the data, such as an edit box, checkbox set, or pop-up menu.

**Field boundary**

In Layout mode, an outline that shows the size of a field. To see field boundaries, choose View menu > Show > Field Boundaries. You can specify in Layout mode that field boundaries appear in other database modes by choosing Format menu > Field Borders. (In Browse mode and Find mode, when you make a field active by clicking in it, you see the field boundary whether or not you specify borders.)

**Field label**

Text on a layout that identifies a field. When you place a field on a layout, you can have FileMaker Pro add a field label that matches the field name. You can change or delete this field label if you want.

**Field name**

The name you assign to a field when you define the field. When you place a field onto a layout, you can have FileMaker Pro also place an editable field label that matches the field name.
Field type
The part of a field definition that determines what kind of data you can enter in the field and the kinds of operations FileMaker Pro can perform with the data. FileMaker Pro can create text, number, date, time, timestamp, container, calculation, and summary fields. (Global fields contain the same value for all records in the database and can be of any type except summary.)

File path
The location of a file in an operating system as identified by the drive, folders, filename, and file extension.

File reference
A named file path that identifies an external FileMaker file, such as a related file, used by the current database file. File references are useful, for example, when you're working with value lists and scripts. In the Define File References dialog box, you can manage named references and define one or more alternative paths to an external file.

FileMaker Network
A communications method built into FileMaker Pro that allows you to share FileMaker Pro files hosted by FileMaker Pro or FileMaker Server with others over a network. The FileMaker Network settings and privileges you set up determine how other users (called "clients") can open and use the shared file.

Find mode
The FileMaker mode in which you specify criteria for finding a subset of records.
(Use Browse mode to enter and edit your information; use Layout mode to design how your information is displayed. Use Find mode to find records that match search criteria; use Preview mode to display how your records will print.)

Find request
In Find mode, a blank form based on the current layout. Enter search criteria into one or more fields of the find request.

Firewall
A security system used to prevent unauthorized users from gaining access to a LAN. A firewall usually has a single computer that is connected to the Internet and all Internet traffic must pass through that computer.

Footer part
Use the footer part for page numbers or dates. This part appears at the bottom of every screen or page (unless you add a Title Footer). You can have only one footer in a layout. A field in the footer displays data from the last record on that page.

Form view
Displays one record at a time. Fields appear on separate lines, with the field label on the left and the field data on the right. Select this view using the View as Form menu option.
Formula
A set of instructions that FileMaker Pro follows to calculate a value used in a field or as the criteria for matching database records.

Found set
The set of records in a table that are made active by a find request. When you find all records, the found set is the entire table.

Full Access privilege set
One of the three pre-defined privilege sets that appear in every file. The Full Access privilege set allows complete read/write access to a file, including making changes to privileges for the file.

Fully qualified name
The complete name of a field or layout, expressed using the format tableName::[field or layout name], where “tableName” is the name of the underlying table occurrence in the relationships graph upon which the field or layout is based. A fully qualified name identifies an exact instance of a field or layout. Because fields and layouts with common names can be based on different tables, FileMaker Pro uses fully qualified names to avoid errors in calculations and scripts.

Function
A predefined, named formula that performs a specific calculation and returns a single, specific value.

Function list separator
The punctuation character (a semicolon) that separates parameters in a function definition. If you type a comma, FileMaker Pro automatically changes it to a semicolon after you close the Specify Calculation dialog box.

G

GIF (Graphics Interchange Format)
A platform-independent file format often used to distribute graphics on the Internet.

Global field
A field defined with the global storage option can contain one value that's used for all records in a file. Use the value of a global field as a fixed value in calculations, to declare variables in If or Loop script steps, or for fields that rarely need to be updated (for example, a company logo in a container field). A global field can be any field type except summary. A global field can't be indexed.

Grand summary
Total or other aggregate value for all records in the found set.
**Grand summary part**
Use grand summary parts to view and display summary information (totals, averages, and so on) in summary fields for all records in the found set. You can add one grand summary part at the top (leading) and one grand summary part at the bottom (trailing) of a layout.

**Grouped object**
A collection of objects that behaves as one object in Layout mode.

**Guest**
A user who opens a protected file without specifying an account name and password. The Guest account is assigned a privilege set that determines what guests can do in the file. Guest access may be disabled for a file.

**Handle**
One of the small squares at the corners of a selected object used to resize and reshape the object.

**Header part**
Use a header part for column headings, titles, and other information that appears only at the top of every page on a layout. FileMaker Pro displays the header in Browse mode and prints it on every page, except the first page if you add a title header. Fields added to a header are printed on every page, using data from the first record on that page.

**Home page**
The starting page for a web site. It often has some form of a table of contents that allows viewers to link to other parts of the web site.

**Host**
After a file has been opened and enabled for sharing, the host is either the first FileMaker Pro user to share the file, or the host is FileMaker Server. Once the host opens the file, other users (clients) can access and change the file. All changes are stored in the file on the hard disk where the file resides. FileMaker Network settings and privileges determine how FileMaker Pro clients interact with databases hosted through FileMaker Pro or FileMaker Server.

**HTML (Hypertext Markup Language)**
A language that is used for displaying and accessing information on the World Wide Web.

**HTTP (Hypertext Transfer Protocol)**
The Internet protocol that defines how a web server responds to requests for files.
Import
To bring (copy) data from a table, another file, or another application into the current table. You can also import scripts from one FileMaker Pro file into another.

Indexing
An option that can be enabled when defining (or changing) the definition of a field. When indexing is enabled, FileMaker Pro builds a list of all the values that occur in the field in the table. This improves the performance of tasks such as finding data, but it increases the size of the database file on disk.

Instant Web Publishing
A method of sharing your FileMaker databases with other users via a web browser. Web clients are limited to the layouts that you provide and to the privilege sets assigned to their user accounts.

To access your database on the web, clients must have the fmiwp extended privilege.

To publish your database on the web, you must have an Internet connection (usually provided through an Internet service provider) and an IP address.

Internet
An international network of many other networks that are linked using the TCP/IP network protocol.

Internet service provider (ISP)
The company from which you purchase your connection to the Internet.

Intranet
A private TCP/IP network of linked computers within a company or organization.

IP (Internet Protocol) address
A four-part number, usually formatted as 12.34.56.78, that uniquely identifies a computer on the Internet.

J, K

JDBC
A Java API that uses SQL statements to access data from, and exchange data with, many database management systems. The JDBC driver communicates between your Java applet and the FileMaker Pro or FileMaker Server Advanced data source.

JPEG (Joint Photographic Experts Group)
A platform-independent file format often used to distribute graphics on the Internet.
Key
A column (or columns) that makes a particular row unique (corresponds to a match field).

L

Labels layout
A predefined layout with fields arranged for printing on mailing label stock, and media and index sheets.

LAN (local area network)
A connection between computers within a location using cable or a wireless system.

Layout
An arrangement of fields, objects, pictures, and layout parts that represents the way information is organized and presented when you browse, preview, or print records. You can design different layouts for entering data, printing reports and mailing labels, displaying web pages, and so on.

Layout mode
The FileMaker mode in which you determine how information in fields is presented on the screen and in printed reports.
(Use Browse mode to enter and edit your information; use Layout mode to design how your information is displayed. Use Find mode to find records that match search criteria; use Preview mode to display how your records will print.)

Layout part
A section of a layout that organizes or summarizes information. Layout parts include Body, Header, Footer, Title Header, Title Footer, leading and trailing Grand Summary, and leading and trailing Subsummary.

Layout pop-up menu
Near the top of the status area, a pop-up menu from which you can choose a defined layout. This menu is available in all modes.
If you don't see the status area, click the status area control at the bottom of the document window.

Layout theme
The appearance of an onscreen or printed report. You can choose a theme when you use the New Layout/Report assistant.
Layout types

FileMaker Pro includes six predefined layout types: Standard Form, Columnar List/Report, Table View, Labels, Envelope, and Blank layout. You can use the predefined layout types as they are, or change them to suit your needs.

To use a predefined layout type, in Layout mode, choose Layouts menu > New Layout/Report. The assistant guides you through creating the type of layout or report you want. To change the layout, use the tools and commands in Layout mode to tailor the layout for your needs.

LDAP (lightweight directory access protocol)

A protocol for accessing online directory services.

Link

On a web page, text or a graphic which -- when you click it -- displays an associated web page or a specific element within a page.

Also, the HTML code that creates a link to another web page or to a specific element within a page.

In OLE, a connection to an object.

Linked OLE object

A linked OLE object exists in an external source file and is displayed in your database. You can update linked objects to work with current data. A linked OLE object can be a graphic object, sound file, movie, or text created in another application. FileMaker Pro starts that other application when you view or work with the object.

List view

Displays records one record at a time in a list format. Select this view using the View as List menu option.

Locked object

An object on a layout that cannot be edited or deleted. To lock an object, in Layout mode, select it and then choose Arrange menu > Lock. The object's selection handles dim.

To unlock an object, choose Arrange menu > Unlock.

Lookup

A lookup matches records and copies data from a related table into a field in the current table. After data is copied, it becomes part of the current table (as well as existing in the table it was copied from). Data copied to a table doesn't automatically change when the values in the related table change.

Lookup target field

The field that you want data copied to during a lookup.

Lookup source field

The field in the related table that contains the data you want copied during a lookup.
Many-to-many relationship

A correspondence between data in database tables in which more than one record in the first table is related to more than one record in another table, and more than one record in that table is related to more than one record in the first table.

Match field

For relational databases and lookups, a field in a source table and a field in a related table that contain values you want to use to find matching records. (A match field is sometimes called a key field or trigger field.)

For importing records, values in the match fields determine which records in the source table update which records in the target table.

Menu

A list of menu items. Each menu has a title that appears on the menu bar.

Menu bar

The area at the top of the screen (Mac OS) or window (Windows) that displays the installed menu set.

Menu item

One item listed in a menu on the menu bar. A menu item corresponds with one command, submenu, or separator.

Menu item properties

All the settings for a menu item, including platform, display title, shortcut, and action.

Menu set

The collection of menus that installs on the menu bar.

Merge field

A placeholder on a layout for the contents of a database field. A merge field expands or contracts in Browse and Preview modes, or when printed, to fit the amount of data in the database field for each record.

Merge fields are useful for mail merge form letters; FileMaker Pro uses merge fields in predefined Labels and Envelope layouts.

Mode

In FileMaker Pro, the four different environments (Browse, Find, Layout, and Preview) that you use to work with your database file.

Mode pop-up menu

A pop-up menu in the lower-left corner of the document window from which you can choose a different mode (Browse, Find, Layout, or Preview). This menu is available in all modes.
Multi-key field
A match field that contains more than one value, each on a separate line. A multi-key field can be used in one table involved in a relationship, to match several possible values in the match field of the other table.

Multimedia
Files that combine media, like text, graphics, sound, animation, and video.

Network protocol
A network protocol (for example, TCP/IP) is a set of rules that govern how computers exchange messages on a network.

New Layout/Report assistant
A wizard that guides you through creating a layout or report according to options you choose.

Object
On a FileMaker Pro layout, an object is a discrete entity or shape that you can select, move, modify, delete, or name. Lines, fields, buttons, portals, imported graphics, blocks of text, tab controls, and web viewers are objects.

Object effect
Gives objects in layouts a 3-dimensional appearance. Choose an object effect from the object effects palette. You can choose Embossed, Engraved, or Drop Shadow effect.

Object Grids
An invisible snap-to grid that aligns objects you create or move.

ODBC
An API that uses SQL statements to access data from, and exchange data with, many database management systems. FileMaker Pro uses ODBC drivers to share data (as a data source) and to interact with data from other applications (as a client application).

OLE client
A document that includes an object linked from another document via OLE (Object Linking and Embedding) or that includes an embedded OLE object.

OLE object
Information from another application that you can include in a FileMaker Pro file. You work with OLE (Object Linking and Embedding) objects -- like graphics, spreadsheets, sounds, or text -- in container fields or layouts in FileMaker Pro files.
1-away relationship
A correspondence between database tables in which two tables are directly related to each other, with no other tables between them.

One-to-many relationship
A correspondence between data in database tables in which one record in the first table is related to more than one record in another table.

One-to-one relationship
A correspondence between data in database tables in which one record in the first table is related to one record in another table.

Operands
Components of a formula. For example, in the formula Quantity * Price, Quantity and Price are the operands.

Operators
In calculations, symbols that indicate how to combine two or more expressions. These include the standard arithmetic operators (+, -, /, *), logical operators that set up conditions that must be met to make a value True or False (AND, OR, XOR, and NOT), and find operators (<, =, @) that help you limit the records defined in a find request.

In the relationships graph, symbols that define the match criteria between one or more pairs of fields in two tables. These include: != (not equal), > (greater than), < (less than), = (equal), <= (less than or equal to), >= (greater than or equal to) and x (all rows, or cartesian product).

P

Parent script
A script that defines script parameters and can call other scripts.

Part label
In Layout mode, the label that appears at the left or side of the bottom dividing line of each layout part. By dragging it up or down, you can use the part label to resize a part. You can also open the Part Definitions dialog box by double-clicking the label.

Plug-in
Software that extends the capabilities of an application in a specific way.

Port
A pre-assigned number that indicates a "logical connection place" where a client (such as a web browser) can connect to a particular server application on a networked computer. Port numbers range from 0 to 65536. Port 80 is the default port for HTTP services such as FileMaker Pro web publishing, but you can use another port number if 80 is already in use by another server application.
Portal
For relational databases, a layout object in one table where you place one or more related fields to display in rows the data from one or more related records.

Preview mode
The FileMaker mode in which you see how layouts will look when they're printed.
(Use Browse mode to enter and edit your information; use Layout mode to design how your information is displayed. Use Find mode to find records that match search criteria; use Preview mode to display how your records will print.)

Privilege set
A defined set of permissions that determines a level of access to a database file. You can define as many privilege sets as you like for a file. There are three pre-defined privilege sets: Full Access, Data Entry Only, and Read-Only Access.

Query
Retrieving, manipulating, or modifying data from a data source by sending SQL statements. Also, requesting, and then receiving, data from a DBMS. You can also add, edit, format, sort, and perform calculations on your data using queries.

QuickTime
An application from Apple Computer, Inc. that compresses, stores, and plays files combining text, sound, animation, and video.

QuickTime VR
A type of QuickTime movie. QuickTime VR movies let you view panoramic images or objects from many angles.

Read-Only Access privilege set
One of the three pre-defined privilege sets that appear in every file. The Read-Only Access privilege set allows read access to the records in a file, but not write or design access.

Record
One set of fields in a database table. Each record contains data about a single activity, individual, subject, or transaction.

Recursive script
A script that calls itself.
Related field
For relational databases, a field in one table that is related to a field in another table (or to a different field within the same table). If a relationship is defined between two tables (even through another table), data in fields in one table can be accessed from the other table.

Related record
A record in the related table whose match field (according to the relationship used) contains a value that's equal to the value in the match field of another table.

Related table
For relational databases, the table that contains the data you want to access and work with in the current table. For lookups, the table that contains the data to copy.

Relational database
A group of one or more database files that, when used together, contain all the data you need. Each occurrence of data is stored in only one table at a time, but can be accessed in any table, either in the same file or from a related file. Data from another table or file is displayed in the current table without being copied, and the data changes whenever the values in the other table or file change.

Relationship
Relationships provide access to data from one table to another. Relationships can join one record in one table to one record in another table, one record to many other records, or all records in one table to all records in another table, depending on the criteria you specify when you create the relationship in the relationships graph.

Relationships graph
In the Relationships tab of the Define Database dialog box, you can see the occurrences of tables both in the current file and from any external, related database files. In this relationships graph, you join tables and change relationships between fields in different tables.

When you create a new table, a visual representation, or occurrence, of the table appears in the relationships graph. You can specify multiple occurrences (with unique names) of the same table in order to work with complex relationships in the graph.

Repeating field
A field containing multiple, separate values.

Report with grouped data
A subsummary report that you create using the Columnar List/Report layout type. Reports with grouped data can include totals and subtotals.

Row
When a database file is viewed as a table, a row corresponds to a record.
S

Schema
In database terminology, a schema is the organization or structure of the elements, objects, and attributes of a database. A schema report will typically list all the database files, fields, scripts, layouts, relationships, tables, and so on in the database, and will document their properties.

Script
One or more instructions (script steps) that you define to automate repetitive or difficult tasks. You define scripts using ScriptMaker. You run a script by clicking its button, choosing its menu command, calling it from another script or a plug-in, or running it at startup or when a file closes.

Script Debugger
A FileMaker Pro Advanced tool for debugging FileMaker Pro scripts.

Script step
A ScriptMaker command that you include in a script.

ScriptMaker
A feature in FileMaker Pro that enables you to create a script that tells FileMaker Pro to perform one action or a sequence of actions, or script steps.

Search criteria
In Find mode, the values and operators you specify to locate records. For example, if you type ABC Travel in the Vendor field, FileMaker Pro looks for and returns all records that have this name in the Vendor field.

Self-join
A relationship between fields in the same table. This creates another occurrence of the table in the relationships graph.

Separator
A line within a menu that separates or groups menu items.

Serial number
A unique number entered by FileMaker Pro for each record. You can tell FileMaker Pro to automatically enter a serial number for each record by setting the Auto-Enter options in the Options for Field dialog box. You can also serialize records by choosing Records menu > Replace Field Contents in Browse mode.
Shared database
A database file for which sharing has been enabled, which permits users to access the database file over a network. FileMaker Pro, FileMaker Pro Advanced, FileMaker Server, and FileMaker Server Advanced each support one or more of the following ways to share databases: FileMaker Network sharing, which permits multiple FileMaker Pro users to use a database file simultaneously; web publishing of databases to web browser users; and sharing of data with other applications via ODBC/JDBC.

Shortcut
Also known as keyboard shortcut. One or more keys that users can press to perform tasks.

Shortcut menu
Use to edit objects or data quickly by choosing commands from a shortcut, or context, menu. Commands vary depending on the mode you’re using, the item the cursor is over, and whether an item is selected.

To display a shortcut menu, Right-click (Windows) or Control-click (Mac OS) the item.

Slider
In the status area, the control below the book for quickly moving to a record in your database file based on its location in the file. For example, in Browse mode, slide the slider to the left to go to the first record and to the right to go to the last record.

In Browse mode, moving the slider changes the current record. In Find mode, moving the slider changes the current find request. In Layout mode, moving the slider changes the current layout. In Preview mode, moving the slider changes the current page.

Sliding objects
Objects that move together to close gaps left by entries in adjacent fields.

Set sliding in Layout mode (choose Format menu > Sliding/Printing).

Sort order
The sequence for rearranging records. Records are sorted by the first field in the sort order list, then the second, and so on. Values within each field are sorted by the order specified (ascending, descending, or custom).

Source file
The file from which you bring data during importing or exporting, or the file from which you add a table to the relationships graph.

Source table
The table upon which one or more tables in the relationships graph are based. The source table is the table defined in the Tables tab of the Define Database dialog box.

SQL
A structured programming query language that controls and interacts with a DBMS.
Stacking order
The order in which objects overlap on a layout. You can change this order by cutting and pasting objects or by choosing Arrange menu > Bring to Front, or Bring Forward, or Send to Back, or Send Backward in Layout mode.

Standard Form layout
The default layout, with all fields arranged on separate lines in the order they were defined. The body part is only as tall as it needs to be to include all the fields in the database. This layout includes header and footer parts.

Startup script
A script that automatically runs when a file is opened. You can script such things as setting system formats to the user's formats or setting a database to be shared in a startup script. You specify a startup script in the File Options dialog box.

Status area
The area at the left side of the document window that displays the book and status information. In Find mode, the status area includes Find buttons. In Layout mode, it includes tools and controls for drawing, and working with fields, layout parts, buttons, and portals.

Submenu
A menu that extends from another menu item.

Sub-script
A script that is called from another script.

Subsummary parts
Use summary parts to view and display information from one or more records. You place a summary field in a summary part to display a summary of information for each group of records sorted on the break field. You can add one or more subsummaries above (leading) or below (trailing) the body.

Subsummary value
Aggregate values for different categories of data within a field. For example, a subsummary value can be the total of employees for each department.

Summary field
A field that contains the result of a summary calculation of values across a group of records.
System formats
Settings you control with control panels to determine how dates, times, currency, and numbers display and sort on your computer. (See the documentation that came with your computer for information on using these control panels.)

If the system formats are different on your computer from the ones on the computer where the database file was created, the first time you open the file, FileMaker Pro will ask you which system formats to use.

T

T square
Nonprinting, movable horizontal and vertical guidelines to help you position and align objects in Layout mode. An object's left or right boundary, top or bottom boundary, or center "snaps to" the T square lines.

Tab control
A layout object made up of one or more tab panels that allows you to organize fields and other objects within each tab panel's borders.

Tab order
The order in which you move from field to field in a record. In Layout mode, you can define a custom tab order and include buttons, tab controls, and web viewers in the tab order.

Tab panel
A component of a tab control. The tab panel is the area displayed when a tab in a tab control is selected. You can place objects such as lines, fields, buttons, portals, imported graphics, blocks of text, tab controls, and web viewers in tab panels.

Table
A collection of data pertaining to a subject, such as customers or stock prices. A database file contains one or more tables, which consist of fields and records. When you create a new table, a visual representation, or occurrence, of the table appears in the relationships graph. You can specify multiple occurrences (with unique names) of the same table in order to work with complex relationships in the graph.

Table view
Displays multiple records in a tabular format like a spreadsheet. Each record appears in a row, and each field appears in a column. Select this view using the View as Table menu option.

Target file
The file into which you bring data during importing.

TCP/IP (Transmission Control Protocol/Internet Protocol)
The basic communication protocol that is the foundation of the Internet.
Template
Or Starter Solution. A pre-designed and formatted FileMaker Pro file, or web page, that you can copy and change for your own use.
Also, a pre-defined website that you can select in the Web Viewer Setup dialog box to help you create a web viewer quickly.

Text baseline
In Layout mode, the dotted guideline that appears at the base of the text in a field or text block. When you move a field or text block, the text baseline extends out horizontally from the object to help you align it with other objects. You can specify in Layout mode that text baselines appear in other database modes by choosing Format menu > Field Borders.

Text expression
Any expression that returns a text result. For example, a text expression can be a constant ("London"), a field reference (Status), or a calculated value (Rightwords(Lastname, 1)).

Timestamp
A field type combining date and time that is compatible with the ODBC requirement for the SQL format [yyyy.mm.dd hh:mm:ss.sss].

Tool panel
In the status area in Layout mode, the collection of tools that includes the selection tool (pointer), text tool, line tool, rectangle tool, rounded rectangle tool, oval tool, field/control tool, portal tool, tab control tool, web viewer tool, and button tool.
If you don't see the status area, click the status area control at the bottom of the document window.

Toolbar
Use items in the toolbar to access many FileMaker Pro commands. Different toolbar items are available in each mode. FileMaker Pro includes four toolbars: Standard, Text Formatting, Arrange, and Tools toolbar.

Tooltip
A small box that displays text when a user moves the cursor over a layout object. Tooltips display in Browse and Find modes.

Unicode
A worldwide standard that, in one code page, provides a unique number for every character in human languages, no matter what the platform, software program, or operating system.
Unit of measure
In Browse and Layout modes, you can set the unit of measure to pixels, inches, or centimeters.

Unstored calculation
A calculation field with a result that is only calculated when the value is needed, for example, to browse or print. In most cases, FileMaker Pro makes a field stored when you define it, but you can change the storage type to unstored.

URL (Uniform Resource Locator)
A web address, which consists of a protocol, a host name, and optionally a port, a directory, and a filename. For example, http://www.filemaker.com/ , ftp://12.34.56.78:80/ myfiles/, or fmp7://mywebsite.com/sample.fp7.

V

Value list
To save time and ensure accuracy during data entry, define frequently used text, number, date, or time values as a value list. When you enter data, you can choose from the list of defined values.

You can format value lists to display in a drop-down list or pop-up menu, or as checkboxes or option (radio) buttons. The values in a value list can be user-defined or based on the values in a field in the same file or in a different file. You can also define relationships for use with value lists, to access and display particular related values. Another option is to use a value list from another file.

Variable
In a calculation, a symbol or name that represents a value. Use the Set Variable script step to specify the name, value, and repetition of the variable. Names prefixed by $ are local variables available only within the current script. Prefix the name with $$ to make the variable available throughout the current file (global). Local and global variables can have the same name but they are treated as different variables.

View
An arrangement of your data primarily useful for onscreen manipulation. In Browse mode, Find mode, or Preview mode, View as Form displays individual records, View as List displays records in a list, and View as Table displays records in a spreadsheet-like table format.

In FileMaker Pro Instant Web Publishing, views are web pages for working with your database in a web browser.
W, X, Y, Z

Web address
The calculated expression that you enter in the Web Viewer Setup dialog box. A web address is not equivalent to a URL that a web user could enter in a web browser.

Web browser
An application that you can use to view web pages/sites on the World Wide Web or an intranet. Browsers download the web pages onto the viewer's computer.

Web page
An HTML document displayed on the Internet or on an intranet.

Web server
A computer that is connected to the Internet or an intranet, and has a web server application installed on it. Web server applications deliver web pages and associated files to web browsers.

Web site
One or more web pages connected by links and displayed on the Internet or on an intranet.

Web user
Someone using a web browser to access a FileMaker Pro database published on the World Wide Web or an intranet.

Web viewer
A layout object that allows you to display information from websites based on data in your database.

World Wide Web
An interlinked collection of web pages residing on web servers, and other documents, menus, and databases, which are available via URLs.

XSLT (Extensible Stylesheet Language Transformations)
XSLT (XSL Transformations) is a subset of XSL (Extensible Stylesheet Language) that is used to transform, or change, the structure of an XML document into a different document format. For example, you can use an XSLT style sheet to transform an XML document into an HTML or TXT document.
XML (Extensible Markup Language)

Instead of being a rigid file format, XML is a language for defining agreed-upon formats that groups can use for exchanging data. Many organizations and businesses are using XML to transfer product information, transactions, inventory, and other business data.

FileMaker Pro can export XML data that can then be used, for example, by spreadsheet applications, data charting applications, and enterprise SQL databases. FileMaker Pro can also import XML data.