Contents

Aggregate functions

Average ................................................................. 16
Count ................................................................. 17
List ................................................................. 18
Max ................................................................. 20
Min ................................................................. 21
StDev ............................................................... 22
StDevP .............................................................. 23
Sum ................................................................. 24
Variance ............................................................. 25
VarianceP .......................................................... 27

Date functions

Date ................................................................. 29
Day ................................................................. 30
DayName ........................................................... 31
DayNameJ .......................................................... 32
DayOfWeek .......................................................... 32
DayOfYear ........................................................... 33
Month .............................................................. 34
MonthName ......................................................... 34
MonthNameJ ......................................................... 35
WeekOfYear .......................................................... 35
WeekOfYearFiscal ................................................... 36
Year ................................................................. 37
YearName ............................................................ 37

Design functions

DatabaseNames ......................................................... 40
FieldBounds .......................................................... 41
FieldComment ......................................................... 42
FieldDs .............................................................. 42
FieldNames .......................................................... 43
FieldRepetitions ....................................................... 44
FieldStyle ........................................................... 44
FieldType ............................................................ 45
GetNextSerialValue .................................................. 46
LayoutIDs ............................................................ 47
LayoutNames ........................................................ 47
LayoutObjectNames ................................................ 48
RelationInfo .......................................................... 49
ScriptDs .............................................................. 50
ScriptNames ........................................................ 50
TableIDs .............................................................. 51
TableNames .......................................................... 51
ValueListIDs ........................................................ 52
ValueListItems ......................................................... 52
ValueListNames ....................................................... 53
External functions

External functions

Financial functions

FV ............................................................ 57
NPV ............................................................ 58
PMT ............................................................ 59
PV ............................................................ 59

Get functions

Get(AccountName) ........................................... 65
Get(ActiveFieldContents) .................................. 66
Get(ActiveFieldName) ....................................... 67
Get(ActiveFieldTableName) ................................. 67
Get(ActiveLayoutObjectName) .............................. 68
Get(ActiveModifierKeys) .................................. 68
Get(ActiveRepetitionNumber) ......................... 69
Get(ActiveSelectionSize) ................................. 70
Get(ActiveSelectionStart) ............................... 70
Get(AllowAbortState) ..................................... 71
Get(AllowToolbarState) ................................. 71
Get(ApplicationLanguage) ............................... 72
Get(ApplicationVersion) ................................. 73
Get(CalculationRepetitionNumber) .................. 73
Get(CurrentDate) .......................................... 74
Get(CurrentHostTimestamp) ............................. 75
Get(CurrentTime) .......................................... 76
Get(CustomMenuSetName) ............................... 77
Get(DesktopPath) ........................................ 78
Get(DocumentsPath) ...................................... 78
Get(DocumentsPathListing) ............................ 79
Get(ErrorCaptureState) ................................. 80
Get(ExtendedPrivileges) ............................... 80
Get(FileMakerPath) ....................................... 81
Get(FileName) ............................................ 81
Get(FilePath) ............................................ 82
Get(FileSize) ............................................. 82
Get(FoundCount) ......................................... 83
Get(HighContrastColor) ................................. 84
Get(HighContrastState) ................................. 85
Get(HostApplicationVersion) ......................... 85
Get(HostIPAddress) ...................................... 86
Get(HostName) ........................................... 87
Get(LastError) ........................................... 87
Get(LastMessageChoice) ............................... 88
Get(LastODBCError) .................................... 89
Get(LayoutChoice) ....................................... 90
Get(LayoutCount) ........................................ 90
Get(LayoutName) ........................................................... 91
Get(LayoutNumber) .................................................... 92
Get(LayoutTableName) .................................................. 92
Get(LayoutViewState) .................................................. 92
Get(MultiUserState) ................................................... 93
Get(NetworkProtocol) ............................................... 93
Get(PageNumber) ...................................................... 94
Get(PageSize) .......................................................... 95
Get(PortalRowNumber) ................................................ 95
Get(PrinterName) ....................................................... 96
Get(PreferencePath) .................................................. 96
Get(PrivilegeSetName) ............................................... 97
Get(RecordAccess) .................................................... 98
Get(RecordID) ........................................................ 99
Get(RecordModificationCount) .................................... 100
Get(RecordNumber) ................................................... 101
Get(RecordOpenCount) ................................................ 101
Get(RecordOpenState) ............................................... 102
Get(RequestCount) ..................................................... 103
Get(RequestOmitState) ............................................... 103
Get(ScreenDepth) .................................................... 104
Get(ScreenHeight) .................................................... 104
Get(ScreenWidth) ..................................................... 105
Get(ScriptName) ....................................................... 106
Get(ScriptParameter) ................................................ 106
Get(ScriptResult) ..................................................... 107
Get(SortState) ........................................................ 108
Get(StatusAreaState) ............................................... 108
Get(SystemDrive) ..................................................... 109
Get(SystemIPAddress) ............................................... 109
Get(SystemLanguage) ............................................... 110
Get(SystemNICAddress) ............................................. 111
Get(SystemPlatform) ................................................ 112
Get(SystemVersion) ................................................... 113
Get(TemporaryPath) ................................................ 113
Get(TextRulerVisible) .............................................. 114
Get(TotalRecordCount) ............................................. 115
Get(TriggerKeystroke) ............................................. 116
Get(TriggerModifierKeys) ....................................... 116
Get(UserCount) ....................................................... 117
Get(UserName) ........................................................ 118
Get(UseSystemFormatState) ....................................... 119
Get(WindowContentHeight) ....................................... 119
Get(WindowContentWidth) ....................................... 120
Get(WindowDesktopHeight) ...................................... 121
Get(WindowDesktopWidth) ...................................... 121
Get(WindowHeight) .................................................. 122
Get(WindowLeft) ...................................................... 123
Get(WindowMode) ..................................................... 123
Get(WindowName) ..................................................... 124
Get(WindowTop) ...................................................... 125
Get(WindowVisible) ................................................ 125
Get(WindowWidth) ................................................... 126
Get(WindowZoomLevel) ............................................ 126
Logical functions
Case ................................................................. 128
Choose ............................................................... 129
Evaluate ............................................................... 130
EvaluationError ..................................................... 131
GetAsBoolean ....................................................... 132
GetField ............................................................... 133
GetFieldName ......................................................... 134
GetLayoutObjectAttribute ........................................ 134
GetNthRecord ....................................................... 136
If ................................................................. 137
IsEmpty ............................................................. 138
IsValid ............................................................. 138
IsValidExpression ................................................ 139
Let ................................................................. 140
Lookup ............................................................... 141
LookupNext ........................................................ 142
Self ................................................................. 143

Number functions
Abs ................................................................. 145
Ceiling ............................................................... 146
Combination ......................................................... 146
Div ................................................................. 147
Exp ................................................................. 148
Factorial ............................................................. 148
Floor ................................................................. 149
Int ................................................................. 149
Lg ................................................................. 150
Ln ................................................................. 150
Log ................................................................. 151
Mod ................................................................. 152
Random ............................................................. 152
Round ............................................................... 153
SetPrecision ......................................................... 154
Sign ................................................................. 154
Sqrt ................................................................. 155
Truncate ............................................................. 155

Repeating functions
Extend ............................................................. 157
GetRepetition ..................................................... 158
Last ................................................................. 158

Summary functions
GetSummary ....................................................... 160

Text functions
Char ................................................................. 163
Code ................................................................. 164
Text formatting functions

RGB ........................................... 194
TextColor ................................... 195
TextColorRemove ............................ 196
TextFont ...................................... 197
TextFontRemove .............................. 198
TextFormatRemove ........................... 199
TextSize ...................................... 199
TextSizeRemove .............................. 200
TextStyleAdd ................................................................. 201
TextStyleRemove ........................................................... 202

Time functions

Hour .................................................................................. 204
Minute ................................................................................. 204
Seconds ............................................................................... 205
Time ................................................................................... 205

Timestamp functions

Timestamp ............................................................................ 207

Trigonometric functions

Acos .................................................................................. 208
Asin ..................................................................................... 209
Atan ..................................................................................... 209
Cos .................................................................................... 210
Degrees ............................................................................... 210
Pi ....................................................................................... 211
Radians ............................................................................... 211
Sin ...................................................................................... 212
Tan ...................................................................................... 212
About functions

Working with formulas and 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.

A formula calculates a single value, based on constants (such as 1.07 or "hello"), operators (such as "+" or ">"), and field references (such as Subtotal or InvoiceTotal) you enter. For example, if sales tax in your area is 7% and you have a field named Subtotal, you could create a field named InvoiceTotal that gets the value of the formula Subtotal * 1.07.

If a formula is especially common or popular, FileMaker Pro gives it a name and defines exactly how you should use it. A named and predefined formula is called a function. For example, if you want to find the average of some test scores, you could write your own formula to add them all and divide by the total number of scores. A simpler approach would be to use the function named Average and follow the rules defined for its use.

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. Links to help topics may not work in this format.

Learning about functions

If you’re new to functions, use the following resources to learn about working with functions.

To use a step-by-step tutorial:

• In FileMaker Pro, choose Help menu > Product Documentation > Tutorial.

To view all help topics about individual functions:

• In FileMaker Pro, choose Help > FileMaker Pro Help. In the help window, choose Reference > Functions reference.

• To view answers to frequently asked questions, tips, troubleshooting advice, and more, visit the FileMaker Knowledge Base at www.filemaker.com/kb/.
Functions reference (alphabetical list)

A, B, C

Abs .......................................................... 145
Acos ...................................................... 208
Asin ...................................................... 209
Atan ...................................................... 209
Average .................................................... 16
Case .......................................................... 128
Ceiling .................................................... 146
Char .......................................................... 163
Choose .................................................... 129
Code .......................................................... 164
Combination ............................................... 146
Cos .......................................................... 210
Count ...................................................... 147

D

DatabaseNames .............................................. 40
Date .......................................................... 29
Day .......................................................... 30
DayName .................................................... 31
DayNameJ ...................................................... 32
DayOfWeek .................................................. 32
DayOfYear ................................................... 33
Degrees ..................................................... 210
Div .......................................................... 147

E

Evaluate ..................................................... 130
EvaluationError ........................................... 131
Exact .......................................................... 165
Exp .......................................................... 148
Extend ....................................................... 157
External ..................................................... 55

F

Factorial ..................................................... 148
FieldBounds ................................................ 41
FieldComment ............................................. 42
FieldDs ...................................................... 42
FieldNames ................................................ 43
FieldRepetitions ......................................... 44
FieldStyle ................................................... 44
FieldType .................................................... 45
Filter .......................................................... 166
FilterValues ............................................... 167
Floor .......................................................... 149
FV ............................................................. 57
Get(AccountName) .......................................................... 65
Get(ActiveFieldContents) ................................................. 66
Get(ActiveFieldName) ......................................................... 67
Get(ActiveTableName) ......................................................... 67
Get(ActiveLayoutObjectName) ............................................. 68
Get(ActiveModifierKeys) ..................................................... 68
Get(ActiveRepetitionNumber) ............................................. 69
Get(ActiveSelectionSize) ............................................... 70
Get(ActiveSelectionStart) ................................................ 70
Get(AllowAbortState) ...................................................... 71
Get(AllowToolBarState) ..................................................... 71
Get(ApplicationLanguage) ............................................... 72
Get(ApplicationVersion) .................................................... 73
Get(CalculationRepetitionNumber) .................................... 73
Get(CurrentDate) .......................................................... 74
Get(CurrentHostTimestamp) ............................................. 75
Get(CurrentTime) .......................................................... 76
Get(CurrentTimestamp) .................................................... 76
Get(CustomMenuSetName) .................................................. 77
Get(DesktopPath) .......................................................... 77
Get(DocumentsPath) ....................................................... 78
Get(DocumentsPathListing) ............................................... 79
Get(ErrorCaptureState) ................................................... 80
Get(ExtendedPrivileges) .................................................. 80
Get(FileMakerPath) ........................................................ 81
Get(FileName) ............................................................. 82
Get(FilePath) ............................................................... 82
Get(FileSize) ............................................................... 83
Get(FoundCount) .......................................................... 83
Get(HighContrastColor) .................................................... 84
Get(HighContrastState) ..................................................... 85
Get(HostApplicationVersion) ........................................... 85
Get(HostName) ............................................................. 85
Get(HostIPAddress) ........................................................ 86
Get(HostApplicationVersion) ........................................... 86
Get(HostName) ............................................................. 87
Get(LastError) ............................................................. 87
Get(LastMessageChoice) ............................................... 88
Get(LastErrorODBCError) ................................................. 89
Get(LayoutAccess) ........................................................ 90
Get(LayoutCount) .......................................................... 90
Get(LayoutName) ........................................................... 91
Get(LayoutNumber) ......................................................... 92
Get(LayoutTableName) ..................................................... 92
Get(LayoutViewState) ...................................................... 93
Get(MultiUserState) ....................................................... 93
Get(NetworkProtocol) ..................................................... 94
Get(PageNumber) .......................................................... 94
Get(PortalRowNumber) .................................................... 95
Get(PreferencesPath) ..................................................... 95
Get(PrinterName) .......................................................... 96
Get(PrivilegeSetName) .................................................... 97
Get(RecordAccess) ........................................................ 97
Get(RecordID) ............................................................. 98
Get(CurrentTimestamp) .................................................... 98
Get(HostName) ............................................................. 99
Get(RecordModificationCount) ................................................................. 100
Get(RecordNumber) .................................................................................. 101
Get(RecordOpenCount) ........................................................................... 101
Get(RecordOpenState) ............................................................................ 102
Get(RequestCount) .................................................................................. 103
Get(RequestOmitState) ............................................................................. 103
Get(ScreenDepth) .................................................................................. 104
Get(ScreenHeight) .................................................................................. 104
Get(ScreenWidth) .................................................................................. 105
Get(ScriptName) ..................................................................................... 106
Get(ScriptParameter) ................................................................................ 106
Get(ScriptResult) .................................................................................... 107
Get(SortState) ......................................................................................... 108
Get(StatusAreaState) ............................................................................... 109
Get(SystemDrive) .................................................................................... 110
Get(SystemIPAddress) ................................................................................ 110
Get(SystemLanguage) ............................................................................. 111
Get(SystemNICAddress) .......................................................................... 112
Get(SystemPlatform) ................................................................................ 112
Get(SystemVersion) ................................................................................ 113
Get(TemporaryPath) ................................................................................ 113
Get(TextRulerVisible) ............................................................................. 114
Get(TotalRecordCount) ........................................................................... 115
Get(TriggerKeystroke) ............................................................................ 116
Get(TriggerModifierKeys) ...................................................................... 116
Get(UserCount) ..................................................................................... 117
Get(UserName) ....................................................................................... 118
Get(UseSystemFormatsState) ................................................................. 119
Get(WindowContentHeight) .................................................................... 119
Get(WindowContentWidth) ..................................................................... 120
Get(WindowDesktopHeight) .................................................................... 121
Get(WindowDesktopWidth) .................................................................... 121
Get(WindowHeight) ................................................................................ 122
Get(WindowLeft) ..................................................................................... 123
Get(WindowMode) .................................................................................. 123
Get(WindowName) .................................................................................. 124
Get(WindowTop) ..................................................................................... 125
Get(WindowVisible) ................................................................................ 125
Get(WindowWidth) .................................................................................. 126
Get(WindowZoomLevel) ........................................................................... 126
GetAsBoolean ......................................................................................... 132
GetAsCSS ............................................................................................... 168
GetAsDate .............................................................................................. 169
GetAsNumber ......................................................................................... 169
GetAsSVG ............................................................................................... 170
GetAsText ............................................................................................... 171
GetAsTime .............................................................................................. 171
GetAsTimestamp .................................................................................... 172
GetAsURLEncoded .................................................................................. 172
GetField ................................................................................................. 133
GetFieldName ........................................................................................ 134
GetLayoutObjectAttribute .................................................................... 134
GetNextSerialValue ................................................................................ 46
GetNthRecord ........................................................................................ 136
GetRepetition .............................................................. 158
GetSummary ............................................................. 160
GetValue ................................................................. 173

H, I, J, K
Hiragana ................................................................. 174
Hour ...................................................................... 204
If ....................................................................... 137
Int ................................................................. 149
IsEmpty ............................................................ 138
IsValid .......................................................... 138
IsValidExpression .................................................. 139
KanaHankaku .................................................... 174
KanaZenkaku ..................................................... 175
KanjiNumeral ..................................................... 175
Katakana ............................................................ 176

L, M, N, O
Last ................................................................. 158
LayoutIDs ......................................................... 47
LayoutNames .................................................... 47
LayoutObjectName .................................................. 48
Left ................................................................. 176
LeftValues ....................................................... 177
LeftWords ......................................................... 177
Length ............................................................. 178
Let ................................................................. 140
Lg ................................................................. 150
List ............................................................... 18
Ln ................................................................. 150
Log ............................................................... 151
Lookup ........................................................... 141
LookupNext ...................................................... 142
Lower ............................................................. 179
Max ............................................................... 20
Middle .......................................................... 179
MiddleValues .................................................. 180
MiddleWords ................................................ 181
Min ............................................................... 21
Minute .......................................................... 204
Mod ............................................................. 152
Month .......................................................... 34
MonthName .................................................... 34
MonthNameJ .................................................. 35
NPV .............................................................. 58
NumToJText ...................................................... 181

P, Q
PatternCount ..................................................... 182
Pi ................................................................. 211
PMT .............................................................. 59
<table>
<thead>
<tr>
<th>Functions Reference (alphabetical list)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
</tr>
<tr>
<td>Proper</td>
</tr>
<tr>
<td>PV</td>
</tr>
<tr>
<td>Quote</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>Radians</td>
</tr>
<tr>
<td>Random</td>
</tr>
<tr>
<td>RelationInfo</td>
</tr>
<tr>
<td>Replace</td>
</tr>
<tr>
<td>RGB</td>
</tr>
<tr>
<td>Right</td>
</tr>
<tr>
<td>RightValues</td>
</tr>
<tr>
<td>RightWords</td>
</tr>
<tr>
<td>RomanHankaku</td>
</tr>
<tr>
<td>RomanZenkaku</td>
</tr>
<tr>
<td>Round</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>ScriptIDs</td>
</tr>
<tr>
<td>ScriptNames</td>
</tr>
<tr>
<td>Seconds</td>
</tr>
<tr>
<td>Self</td>
</tr>
<tr>
<td>SerialIncrement</td>
</tr>
<tr>
<td>SetPrecision</td>
</tr>
<tr>
<td>Sign</td>
</tr>
<tr>
<td>Sin</td>
</tr>
<tr>
<td>Sqrt</td>
</tr>
<tr>
<td>StDev</td>
</tr>
<tr>
<td>StDevP</td>
</tr>
<tr>
<td>Substitute</td>
</tr>
<tr>
<td>Sum</td>
</tr>
<tr>
<td>T, U</td>
</tr>
<tr>
<td>TableIDs</td>
</tr>
<tr>
<td>TableName</td>
</tr>
<tr>
<td>Tan</td>
</tr>
<tr>
<td>TextColor</td>
</tr>
<tr>
<td>TextColorRemove</td>
</tr>
<tr>
<td>TextFont</td>
</tr>
<tr>
<td>TextFontRemove</td>
</tr>
<tr>
<td>TextFormatRemove</td>
</tr>
<tr>
<td>TextSize</td>
</tr>
<tr>
<td>TextSizeRemove</td>
</tr>
<tr>
<td>TextStyleAdd</td>
</tr>
<tr>
<td>TextStyleRemove</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Timestamp</td>
</tr>
<tr>
<td>Trim</td>
</tr>
<tr>
<td>TrimAll</td>
</tr>
<tr>
<td>Truncate</td>
</tr>
</tbody>
</table>

FILEMAKER FUNCTIONS REFERENCE
Upper ........................................................................................................... 192

**V, W, X, Y, Z**

ValueCount .................................................................................................... 192
ValueListIDs .................................................................................................. 52
ValueListItems ............................................................................................... 53
ValueListNames ............................................................................................... 53
Variance ........................................................................................................... 25
VarianceP ......................................................................................................... 27
WeekOfYear ..................................................................................................... 35
WeekOfYearFiscal ......................................................................................... 36
WindowNames ................................................................................................. 54
WordCount ...................................................................................................... 193
Year .................................................................................................................. 37
YearName ......................................................................................................... 37
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`.

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><strong>Average</strong></td>
<td>The average of all valid, non-blank values in the specified field.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>The number of valid, non-blank values in the specified field.</td>
</tr>
<tr>
<td><strong>List</strong></td>
<td>The concatenation of all non-blank values in list form, separated by carriage returns.</td>
</tr>
<tr>
<td><strong>Max</strong></td>
<td>The highest valid value in a field or fields.</td>
</tr>
<tr>
<td><strong>Min</strong></td>
<td>The smallest valid non-blank value in a field or fields.</td>
</tr>
<tr>
<td><strong>StDev</strong></td>
<td>The standard deviation of a series of valid non-blank values in a field or fields.</td>
</tr>
<tr>
<td><strong>StDevP</strong></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><strong>Sum</strong></td>
<td>The total of all valid, non-blank values in the specified fields.</td>
</tr>
<tr>
<td><strong>Variance</strong></td>
<td>The variance of a series of valid non-blank values in a field or fields.</td>
</tr>
<tr>
<td><strong>VarianceP</strong></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**

**Purpose**

Returns a value that is the average of all valid, non-blank values in field.

**Format**

`Average(field{;field...})`

**Parameters**

`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**

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, \( \frac{1+5+6}{3}=4; \frac{2+6}{2}=4; \frac{7}{1}=7; \frac{8}{1}=8 \).

**Count**

**Purpose**

Returns the number of valid, non-blank values in field.

**Format**

`Count(field{;field...})`
Parameters

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

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

Purpose

Returns a concatenated list of non-blank values (separated by carriage returns) for a field or fields.

Format

List(field{;field...})
Parameters
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
Use this function to return a list of values for:
- a single field (table::field), which returns 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...), which returns 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 values across the first repetition in the calculation's first repetition, then the list of 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**

**Purpose**
Returns the highest valid value in field.

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

**Parameters**
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**
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.
Aggregate functions

- 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

Purpose
Returns the smallest valid non-blank value in field.

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

Parameters
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
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
  \(\text{table::field1;table::field2;...}\). You can include fields from different tables
  \(\text{table 1::field A;table 2::field B...}\).

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

\(\min(\text{Bids::Price})\) 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.

\(\min(\text{Field2})\) returns 5 when the calculation isn’t a repeating field.
\(\min(\text{Field1};\text{Field2};\text{Field3})\) 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

Purpose
Returns the standard deviation of the sample represented by a series of non-blank values in field.

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

Parameters
field - any related, 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
Field can be any of the following:
• 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 \(\text{field1;field2;field3}\).
• corresponding repetitions of repeating fields in a record \(\text{repeatingField1;repeatingField2;repeatingField3}\), if the result is returned in a repeating field with at least the same number of repeats.
Aggregate functions

• 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...).

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

Examples

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

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

StDev(Field1;Field2;Field3) returns 2.64575131..., 3.05505046..., 2.12132034..., 2.82842712...

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

StDevP

Purpose

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

Format

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

Parameters

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

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...).

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

**Examples**

A portal displays the related values 5, 6, 7, and 8 in the field Scores. StDevP(table::Scores) 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.

StDevP(Field4) results in an error because the population standard deviation of a single number is not defined.

StDevP(Field2) returns 1.11803398... for a non-repeating field.

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

**Note** 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.

**Sum**

**Purpose**
Returns the total of all valid, non-blank values in field.

**Format**
Sum(field{;field...})

**Parameters**
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.
Aggregate functions

Data type returned
number

Description
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
An Invoice table has a portal showing line items.
Sum(LineItems::ExtendedPrice) 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.
Sum(Hours::BillableHours) 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:

- Sum(Field2) returns 26.
- Sum(Field1;Field2;Field3) returns 12.

If the calculation result is a repeating field:

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

Note When a referenced field is a repeating field, the Sum function returns the sum of the first repetition field, then the sum of the second repetition field, and so on.

Variance

Purpose
Returns the variance of a sample represented by a series of non-blank values in field.
Aggregate functions

Variance(field{;field...})

Parameters
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
The variance of a distribution is a measure of how spread out the distribution is. 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...).

\[
\text{Variance} = \frac{x_1^2 + x_2^2 + ... + x_n^2}{n - 1} - \frac{(x_1 + x_2 + ... + x_n)^2}{n(n - 1)}
\]

Examples
A portal displays the related values 5, 6, 7, and 8 in Scores.
Variance(table::Scores) returns 1.66666666....

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.

Variance(Field4) results in an error since the variance of a single value is not defined.

Variance(Field1;Field2;Field3) returns 7, 9.33333333..., 4.5, 8 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 variance for each class is:
Class 1: 3.80952380...
Class 2: 361.90476190...
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

Purpose
Returns the variance of a population represented by a series of non-blank values in field.

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

Parameters
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
The variance of a population distribution is a measure of how spread out the distribution is. 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...).

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

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

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

VarianceP(Field1;Field2;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.
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 Conversion of dates with two-digit years.

**Note** System formats affect the way dates are displayed. See Opening files with foreign system formats.

**Tip** You can use zero (0) and negative numbers as Date function arguments. For example, the following formula returns 5/31/2010:

\[
\text{Date}(6;0;2010)
\]

Click a function name for details.

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

**Date**

**Purpose**

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

**Format**

\[
\text{Date}(\text{month};\text{day};\text{year})
\]
**Parameters**

- **month** - the month of the year (a two-digit number; see note).
- **day** - the day of the month (a two-digit number; see note).
- **year** - the year (four digits between 0001 and 4000. For example, 2010 but not 10).

**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**

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 your operating system.

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 [Conversion of dates with two-digit years](#).

**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. The date function also allows zero and negative numbers as parameters.

**Examples**

- `Date(10;10;2010)` returns **10/10/2010**.
- `Date(13;1;2010)` returns **1/1/2010** (one month after December 1, 2009).
- `Date(6;0;2010)` returns **5/31/2010** (one day before June 1, 2010).
- `Date(6;-2;2010)` returns **5/29/2010** (three days before June 1, 2010).

**Day**

**Purpose**

Returns a number in the range 1 through 31, representing the day of the month on which `date` occurs.

**Format**

`Day(date)`
Parameters
date - any calendar date

Data type returned
number

Description
Use Day, for example, to identify the day of the month on which 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 Conversion of dates with two-digit years.

Examples
Day("5/15/2010") 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

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

Format
DayName(date)

Parameters
date - any calendar date

Data type returned
text

Examples
DayName(Date(10;5;2010)) returns Tuesday.
DayName(ProjectDue) returns Tuesday when ProjectDue is 10/5/2010.
DayName("10/5/2010") returns Tuesday.
"Return your selection by “ & DayName(DueDate) displays the text Return your selection by followed by the name of the day stored in DueDate.
**Important** To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see [Conversion of dates with two-digit years](#).

---

**DayNameJ**

**Purpose**

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

**Format**

`DayNameJ(date)`

**Parameters**

`date` - any calendar date

**Data type returned**

text

**Examples**

`DayNameJ(Date(4;2;2008))` returns **金曜日**

---

**DayOfWeek**

**Purpose**

Returns a number representing the day of the week that `date` falls on.

**Format**

`DayOfWeek(date)`

**Parameters**

`date` - any calendar date

**Data type returned**

number
**Description**

The number 1 represents Sunday, 2 represents Monday, 3 represents Tuesday, and so on. For example, you can find out on what day of the week a holiday 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 Conversion of dates with two-digit years.

**Examples**

DayOfWeek(Date(10;7;2010)) returns 5.
DayOfWeek(ProjectDue) returns 3 when the date in ProjectDue is 10/5/2010.

**DayOfYear**

**Purpose**

Returns a number equal to the number of days from the beginning of the year of date.

**Format**

DayOfYear(date)

**Parameters**

date - any calendar date

**Data type returned**

number

**Examples**

DayOfYear(Billing Date) returns 32, when Billing Date is 2/1/2010.

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)

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

**Purpose**
Returns a number in the range 1 through 12, representing the number of the month of the year in which `date` occurs.

**Format**
Month(date)

**Parameters**
date - any calendar date

**Data type returned**
number

**Examples**
Month("3/19/2010") 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 19, 2010. (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.

---

**Important**
To avoid errors when using dates, always use four-digit years. For more information about how FileMaker Pro handles two-digit dates, see [Conversion of dates with two-digit years](#).

---

MonthName

**Purpose**
Returns the full name of the month for `date`.

**Format**
MonthName(date)

**Parameters**
date - any calendar date

**Data type returned**
text

**Examples**
MonthName("6/6/2010") 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 4/4/2010.

“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.

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

---

### MonthNameJ

**Purpose**

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

**Format**

`MonthNameJ(date)`

**Parameters**

date - any calendar date

**Data type returned**

text

**Examples**

`MonthNameJ(“6/6/2010”)` returns 6 ⦿

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

---

### WeekOfYear

**Purpose**

Returns the number of weeks after January 1 of the year of `date`.

**Format**

`WeekOfYear(date)`

**Parameters**

date - any calendar date
Data type returned

number

Description

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 Conversion of dates with two-digit years.

Examples

WeekOfYear("1/1/2010") returns 1.
WeekOfYear(ProjectDue) returns 6, when ProjectDue is 2/4/2010.

WeekOfYearFiscal

Purpose

Returns a number between 1 and 53 representing the week containing date, figured according to startingDay.

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

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 2008 you selected Sunday (1) as the starting date, then January 1, 2, or 3 in 2009 would occur in week 53 of fiscal year 2008 (in 2009, January 1 is on a Thursday). The first day of fiscal year 2009 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 [Conversion of dates with two-digit years](#).

## Examples

```
WeekOfYearFiscal(Date(1;2;2008);1) returns 53.
WeekOfYearFiscal(Date(1;7;2008);1) returns 1.
WeekOfYearFiscal(Date(1;1;2009);5) returns 1.
```

## Year

### Purpose

Returns a number representing the year in which `date` occurs.

### Format

`Year(date)`

### Parameters

- `date` - any calendar date

### Data type returned

number

### Description

You can, for example, extract the year from a field containing the date an item was 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 [Conversion of dates with two-digit years](#).

### 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

### Purpose

Returns the Japanese year name of `date`, provided in the specified format.
**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**

If the value for format is blank or other than 0, 1, or 2, then 0 is used.

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

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

**Examples**

`YearName(DateField;0)` Returns 平成20 when DateField contains 7/15/2008.
Design functions

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 [Creating and managing privilege sets](#) 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”
- layouts 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><strong>DatabaseNames</strong></td>
<td>A list of the names of all database files open on the computer (including files opened as a client), separated by carriage returns.</td>
</tr>
<tr>
<td><strong>FieldBounds</strong></td>
<td>The location of each side of the specified field and its rotation in degrees.</td>
</tr>
<tr>
<td><strong>FieldComment</strong></td>
<td>The specified field’s comment.</td>
</tr>
<tr>
<td><strong>FieldIDs</strong></td>
<td>A list of all field IDs in the specified database file and layout, separated by carriage returns.</td>
</tr>
<tr>
<td><strong>FieldNames</strong></td>
<td>A list of the names of all fields on the specified layout, separated by carriage returns.</td>
</tr>
<tr>
<td><strong>FieldRepetitions</strong></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><strong>FieldStyle</strong></td>
<td>The formatting applied to the specified field on the specified layout.</td>
</tr>
<tr>
<td><strong>FieldType</strong></td>
<td>Information about the specified field.</td>
</tr>
<tr>
<td><strong>GetNextSerialValue</strong></td>
<td>The next serial number for the specified field in the specified database file.</td>
</tr>
<tr>
<td><strong>LayoutIDs</strong></td>
<td>A list of all layout IDs in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td><strong>LayoutNames</strong></td>
<td>A list of the names of all layouts in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td><strong>LayoutObjectNames</strong></td>
<td>A list of the names of all named layout objects, separated by carriage returns.</td>
</tr>
<tr>
<td><strong>RelationInfo</strong></td>
<td>A list of four values for each relationship directly related to the specified table.</td>
</tr>
<tr>
<td><strong>ScriptIDs</strong></td>
<td>A list of all script IDs in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td><strong>ScriptNames</strong></td>
<td>A list of the names of all scripts in the specified database file, separated by carriage returns.</td>
</tr>
<tr>
<td><strong>TableIDs</strong></td>
<td>A list of all table IDs in the specified database file, separated by carriage returns.</td>
</tr>
</tbody>
</table>
### DatabaseNames

**Purpose**
Returns a list of the names of all database files open on the computer, separated by carriage returns.

**Format**
DatabaseNames

**Parameters**
None

**Data type returned**
text

**Description**
The names returned 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.

**Examples**
To determine whether Customers is one of the files currently open, use the `DatabaseNames` function with the `FilterValues` function in the formula:

```
FilterValues( DatabaseNames;"Customers")
```

If the formula returns any text value, then Customers is open.

If you want to know how many files with the same name are open, use the `DatabaseNames` function with the `PatternCount` function in the formula:

```
PatternCount(FilterValues(DatabaseNames;"Customers");"Customers")
```

This will tell you how many files named Customers are open.
FieldBounds

**Purpose**
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.

**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](#) for information about literal text parameters.

**Data type returned**
text

**Description**
The location returned 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 counter-clockwise 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.

**Examples**
`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

**Purpose**
Returns the specified field’s comment.

**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](#) for information about literal text parameters.

**Data type returned**
text

**Description**
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

**Purpose**
Returns a list of all field IDs in `fileName` and `layoutName`, separated by carriage returns.

**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](#) for information about literal text parameters.
**Data type returned**

text

**Description**

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**

**Purpose**

Returns a list of the names of all fields on `layoutName`, in `fileName` file, separated by carriage returns.

**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](#) for information about literal text parameters.

**Data type returned**

text

**Description**

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 **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

Purpose

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.

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 for information about literal text parameters.

Data type returned
text

Description

If fieldName isn't a repeating field, it returns 1 vertical.

Examples

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

Purpose

Returns the field formatting applied to fieldName on layoutName in the fileName file.
**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 for information about literal text parameters.

**Data type returned**

text

**Description**

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 Populist.
- A pop-up menu returns Popupmenu.
- A checkbox returns Checkbox.
- A radio button returns RadioButton.
- A drop-down calendar returns Calendar.

**Examples**

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**

**Purpose**

Returns information about fieldName.

**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 for information about literal text parameters.

Data type returned
text

Description
Field names must be in the format `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 three 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

Purpose
Returns the next serial number of `fieldName` in `fileName`.

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 for information about literal text parameters.
Data type returned

text

Description

Field names must be fully qualified in the format tablename::fieldname to specify a field that exists in a table different from the current table.

Examples

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

LayoutIDs

Purpose

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

Format

LayoutIDs(fileName)

Parameters

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

Important  See Design functions for information about literal text parameters.

Data type returned

text

Examples

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

LayoutNames

Purpose

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

Format

LayoutNames(fileName)

Parameters

fileName - the name of an open database file (local or remote).
Important  See Design functions for information about literal text parameters.

Data type returned
text

Examples
LayoutNames("Customers") returns a list of all the layouts in the Customers database file.

LayoutObjectNames

Purpose
Returns a list of the names of all named objects on layoutName in fileName, separated by carriage returns.

Format
LayoutObjectNames(fileName;layoutName)

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

Important  See Design functions for information about literal text parameters.

Data type returned
text

Description
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.

Examples
LayoutObjectNames ("Customers";"Data Entry") returns a list of named objects in the Customers database file that appear on the Data Entry layout.
RelationInfo

**Purpose**
Returns a list of four values for each relationship directly related to `tableName`.

**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 [Design functions](#) for information about literal text parameters.

**Data type returned**
text

**Description**
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**: Data Source Name of the database table connected to `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`.

**Examples**
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**

**Purpose**
Returns a list of all script IDs in `fileName`, separated by carriage returns.

**Format**
```
ScriptIDs(fileName)
```

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

**Important** See Design functions for information about literal text parameters.

**Data type returned**
text

**Examples**
```
ScriptIDs("Customers") returns a list of all the script IDs in the Customers database file.
```

**ScriptNames**

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

ScriptNames(fileName)

**Parameters**

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

---

**Important** See [Design functions](#) for information about literal text parameters.

---

**Data type returned**

text

**Examples**

ScriptNames("Customers") returns a list of all the scripts in the Customers database file.

---

**TableIDs**

**Purpose**

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

**Format**

TableIDs(fileName)

**Parameters**

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

---

**Important** See [Design functions](#) for information about literal text parameters.

---

**Data type returned**

text

**Description**

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.

**Examples**

TableIDs("University Database") returns

1065089
1065090

for the University Database database file if two tables have been defined for the file.
TableNames

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

**Format**
`TableNames(fileName)`

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

**Important** See Design functions for information about literal text parameters.

**Data type returned**
text

**Examples**
- `TableNames("University Database")` returns table occurrences Teacher and Coaches
- for the University Database database file if a Teachers table and a Coaches table have been defined for the file.

ValueListIDs

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

**Format**
`ValueListIDs(fileName)`

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

**Important** See Design functions for information about literal text parameters.

**Data type returned**
text
Examples

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

ValueListItems

Purpose

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

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.

Important  See Design functions for information about literal text parameters.

Data type returned
text

Examples

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

ValueListNames

Purpose

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

Format

ValueListNames(fileName)

Parameters

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

Important  See Design functions for information about literal text parameters.
Data type returned

text

Examples

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

WindowNames

Purpose

Returns a list of the names of windows that are currently open.

Format

WindowNames{(fileName)}

Parameters

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

Parameters in curly braces {} are optional.

Data type returned

text

Description

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, and 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.
External functions

Use external functions to access FileMaker Pro plug-ins. Plug-ins add features to FileMaker Pro. For more information, see Setting plug-in preferences.

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><strong>External</strong></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 Updating plug-ins.

External

**Purpose**

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.

**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
Plug-ins created for FileMaker Pro version 7.0 and later do not use the `External("function name", parameter)` syntax. For more information, see External functions and the FileMaker Pro Advanced Development Guide included with FileMaker Pro Advanced.
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 \texttt{PMT} function.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{FV}</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>\texttt{NPV}</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>\texttt{PMT}</td>
<td>The payment required by the term, interest rate, and principal.</td>
</tr>
<tr>
<td>\texttt{PV}</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>

\textbf{FV}

\textbf{Purpose}

Returns the future value (\texttt{FV}) of an initial investment, based on a constant interest rate and payment amount for the number of periods in months.

\textbf{Format}

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

\textbf{Parameters}

- \texttt{payment} - payment to be made per period
- \texttt{interestRate} - interest rate per period
- \texttt{periods} - number of periods

\textbf{Data type returned}

number

\textbf{Description}

Use this function to calculate \texttt{FV}. 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.

\textbf{Notes}

- When \texttt{interestRate} is 0, this function returns the result of \texttt{payment} * periods.
- The \texttt{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.

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

Examples

FV(50; .11/12; 5 * 12) returns 3975.90398429...

FV(2000; .12; 30) + 5000 * (.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

Purpose

Returns the net present value (NPV) of a series of unequal payments made at regular intervals, assuming a fixed interestRate per interval.

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

Use this function to calculate NPV. For example, if someone borrows money from you and pays you back in unequal amounts over a period of several years, you can use the NPV function to calculate the result.

NPV = \frac{\text{loan amount}}{1 + \text{interestRate}} + \frac{\text{first payment}}{(1 + \text{interestRate})^2} + \frac{\text{second payment}}{(1 + \text{interestRate})^3} + \ldots + \frac{\text{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

**Purpose**
Returns the payment (PMT) required by the term, interestRate, and principal.

**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**
Use this function to calculate PMT.

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

**Examples**
In the following example, the PMT function calculates payments for purchasing a sports car costing $21,000, at an annual rate of 6.9% over 48 monthly payments.
PMT(21000;.069/12;48) returns the payment amount $501.90.
PMT(Cost;.13;Years) returns a payment amount, based on the purchase value stored in Cost, at a 13 percent rate, over the duration stored in Years.
"Your payment will be " & PMT(150000;.13/12;Months) & "." returns Your payment will be, followed by the payment amount, based on a total cost of $150,000, at a 13 percent annual percentage rate, over the duration stored in Months.

PV

**Purpose**
Returns the present value (PV) of a series of equal payments made at regular intervals (periods), assuming a fixed interestRate per interval.

**Format**
PV(payment;interestRate;periods)
Financial functions

Parameters

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

Data type returned

number

Description

Use this function to calculate $PV$.

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

Note  When interestRate is 0, this function returns the result of payment * periods.

Examples

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 percent annually, with the following entry you could find out what those payments are worth with the $PV$ function.

$PV(500; .05; 5)$ returns $2164.73833531...$

If you want the return value to return two decimal places, enclose the formula with the correct Round function: Round(Current Formula; 2).
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 choose View all functions by name, you see only Get(flag).

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Click a function name for details.

<table>
<thead>
<tr>
<th>This function</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get(AccountName)</td>
<td>The authenticated account name being used for the active database file.</td>
</tr>
<tr>
<td>Get(ActiveFieldContents)</td>
<td>The contents of the field that has the focus.</td>
</tr>
<tr>
<td>Get(ActiveFieldName)</td>
<td>The name of the field that has the focus.</td>
</tr>
<tr>
<td>Get(ActiveFieldTableName)</td>
<td>The name of the table that contains the active field (the field that has the focus).</td>
</tr>
<tr>
<td>Get(ActiveLayoutObjectName)</td>
<td>The name of the active layout object in the calculation's active window.</td>
</tr>
<tr>
<td>Get(ActiveModifierKeys)</td>
<td>A number representing the keyboard modifier keys (for example, Shift) that are being pressed.</td>
</tr>
<tr>
<td>Get(ActiveRepetitionNumber)</td>
<td>A number representing the active repetition of a repeating field (the repetition that has the focus).</td>
</tr>
<tr>
<td>Get(ActiveSelectionSize)</td>
<td>A number representing how many characters are selected.</td>
</tr>
<tr>
<td>Get(ActiveSelectionStart)</td>
<td>A number representing the starting character of the selected text.</td>
</tr>
<tr>
<td>Get(AllowAbortState)</td>
<td>A Boolean value representing the current state of Allow user abort script step.</td>
</tr>
<tr>
<td>Get(AllowToolbarState)</td>
<td>A Boolean value representing whether toolbars are allowed to be visible.</td>
</tr>
<tr>
<td>Get(ApplicationLanguage)</td>
<td>Text representing the current application language (for example, English).</td>
</tr>
<tr>
<td>Get(ApplicationVersion)</td>
<td>Text representing the FileMaker Pro application version.</td>
</tr>
<tr>
<td>Get(CalculationRepetitionNumber)</td>
<td>A number representing the repetition of the calculation field that is currently being calculated.</td>
</tr>
<tr>
<td>Get(CurrentDate)</td>
<td>The current date according to the system calendar.</td>
</tr>
<tr>
<td>Get(CurrentHostTimestamp)</td>
<td>The host's current date and time (to the nearest second) according to the system clock.</td>
</tr>
<tr>
<td>This function</td>
<td>Returns</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Get(CurrentTime)</td>
<td>The current time (to the nearest second) according to the system clock.</td>
</tr>
<tr>
<td>Get(CurrentTimestamp)</td>
<td>The current date and time (to the nearest second) according to the system clock.</td>
</tr>
<tr>
<td>Get(CustomMenuSetName)</td>
<td>The name of the active custom menu set.</td>
</tr>
<tr>
<td>Get(DesktopPath)</td>
<td>The path to the desktop folder for the current user.</td>
</tr>
<tr>
<td>Get(DocumentsPath)</td>
<td>The path to the Documents folder for the current user.</td>
</tr>
<tr>
<td>Get(DocumentsPathListing)</td>
<td>A list of all the files and folders in the Documents folder returned by the Get(DocumentsPath) function.</td>
</tr>
<tr>
<td>Get(ErrorCaptureState)</td>
<td>A Boolean value representing the state of Error capture script step.</td>
</tr>
<tr>
<td>Get(ExtendedPrivileges)</td>
<td>A list of keywords for the enabled extended privileges, separated by carriage returns</td>
</tr>
<tr>
<td>Get(FileMakerPath)</td>
<td>The path to the folder of the currently running copy of FileMaker Pro.</td>
</tr>
<tr>
<td>Get(FileName)</td>
<td>The name of the currently active database file.</td>
</tr>
<tr>
<td>Get(FilePath)</td>
<td>The full path indicating the location of the file.</td>
</tr>
<tr>
<td>Get(FileSize)</td>
<td>The size (in bytes) of the currently active database file.</td>
</tr>
<tr>
<td>Get(FoundCount)</td>
<td>A number that represents the number of records in the current found set.</td>
</tr>
<tr>
<td>Get(HighContrastColor)</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)</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(HostApplicationVersion)</td>
<td>The version of FileMaker Pro or FileMaker Server running on the computer that is hosting the current database.</td>
</tr>
<tr>
<td>Get(HostIPAddress)</td>
<td>The IP address of the host machine for the current database.</td>
</tr>
<tr>
<td>Get(HostName)</td>
<td>The registered name of the computer that is hosting the database file.</td>
</tr>
<tr>
<td>Get(LastError)</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)</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)</td>
<td>A string that shows the error state published by ODBC standards, based on ISO/IEF standards.</td>
</tr>
<tr>
<td>Get(LayoutAccess)</td>
<td>A number corresponding to the layout access privileges assigned through the Manage Accounts &amp; Privileges dialog box.</td>
</tr>
<tr>
<td>Get(LayoutCount)</td>
<td>The total number of layouts in the database file.</td>
</tr>
<tr>
<td>Get(LayoutName)</td>
<td>The name of the layout currently displayed.</td>
</tr>
<tr>
<td>Get(LayoutNumber)</td>
<td>The number of the layout currently displayed, according to the list in the Manage Layouts dialog box.</td>
</tr>
<tr>
<td>Get(LayoutTableName)</td>
<td>The name of the table that the layout is displaying records from.</td>
</tr>
<tr>
<td><strong>This function</strong></td>
<td><strong>Returns</strong></td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Get(LayoutViewState)</code></td>
<td>Information about how the database file is being viewed.</td>
</tr>
<tr>
<td><code>Get(MultiUserState)</code></td>
<td>A number representing the current multi-user state of the database file.</td>
</tr>
<tr>
<td><code>Get(NetworkProtocol)</code></td>
<td>The name of the network protocol that FileMaker Pro is using on this machine.</td>
</tr>
<tr>
<td><code>Get(PageNumber)</code></td>
<td>A number representing the current page being printed or previewed.</td>
</tr>
<tr>
<td><code>Get(PortalRowNumber)</code></td>
<td>The number of the portal row containing the focus.</td>
</tr>
<tr>
<td><code>Get(PreferencesPath)</code></td>
<td>The path to the preferences folder for the current user.</td>
</tr>
<tr>
<td><code>Get(PrivilegeSetName)</code></td>
<td>The name of the privilege set being used by the current user.</td>
</tr>
<tr>
<td><code>Get(RecordAccess)</code></td>
<td>A number indicating the access privileges of the current record.</td>
</tr>
<tr>
<td><code>Get(RecordID)</code></td>
<td>The unique ID number of the current record.</td>
</tr>
<tr>
<td><code>Get(RecordModificationCount)</code></td>
<td>The total number of times changes to the current record have been committed.</td>
</tr>
<tr>
<td><code>Get(RecordNumber)</code></td>
<td>The number of the current record in the current found set.</td>
</tr>
<tr>
<td><code>Get(RecordOpenCount)</code></td>
<td>The total number of open records in the current found set that haven't yet been saved.</td>
</tr>
<tr>
<td><code>Get(RecordOpenState)</code></td>
<td>A number representing the state of the current record.</td>
</tr>
<tr>
<td><code>Get(RequestCount)</code></td>
<td>The total number of find requests currently defined for the current table.</td>
</tr>
<tr>
<td><code>Get(RequestOmitState)</code></td>
<td>A Boolean value representing the state of the <strong>Omit</strong> checkbox in Find mode.</td>
</tr>
<tr>
<td><code>Get(ScreenDepth)</code></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><code>Get(ScreenHeight)</code></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><code>Get(ScreenWidth)</code></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><code>Get(ScriptName)</code></td>
<td>The name of the script currently running (or paused).</td>
</tr>
<tr>
<td><code>Get(ScriptParameter)</code></td>
<td>The script parameter passed into the current script.</td>
</tr>
<tr>
<td><code>Get(ScriptResult)</code></td>
<td>The script result from a performed subscript.</td>
</tr>
<tr>
<td><code>Get(SortState)</code></td>
<td>A number value representing the current sort state.</td>
</tr>
<tr>
<td><code>Get(StatusAreaState)</code></td>
<td>A number representing whether the status toolbar is hidden, visible, visible and locked, or hidden and locked.</td>
</tr>
<tr>
<td><code>Get(SystemDrive)</code></td>
<td>The drive letter (Windows) or the volume name (Mac OS) where the currently running operating system is located.</td>
</tr>
<tr>
<td><code>Get(SystemIPAddress)</code></td>
<td>The IP addresses of all the machines connected to a NIC (Network Interface Controller) card.</td>
</tr>
<tr>
<td><code>Get(SystemLanguage)</code></td>
<td>The language currently set on the current system.</td>
</tr>
<tr>
<td><code>Get(SystemNICAddress)</code></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)</td>
<td>A number indicating the current platform.</td>
</tr>
<tr>
<td>Get(SystemVersion)</td>
<td>The version of the operating system of the machine on which the function is executed.</td>
</tr>
<tr>
<td>Get(TemporaryPath)</td>
<td>The path to the current user’s temporary folder used by FileMaker Pro.</td>
</tr>
<tr>
<td>Get(TextRulerVisible)</td>
<td>A Boolean value representing whether or not the text ruler is visible.</td>
</tr>
<tr>
<td>Get(TotalRecordCount)</td>
<td>The total number of records in the current table.</td>
</tr>
<tr>
<td>Get(TriggerKeystroke)</td>
<td>A string containing the characters that activated an OnObjectKeystroke or OnLayoutKeystroke script trigger.</td>
</tr>
<tr>
<td>Get(TriggerModifierKeys)</td>
<td>The state of the keyboard modifier keys as they were when the script trigger was activated.</td>
</tr>
<tr>
<td>Get(UserCount)</td>
<td>The number of users who are currently accessing the file.</td>
</tr>
<tr>
<td>Get(UserName)</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(UseSystemFormatState)</td>
<td>A Boolean value representing the state of the Use System Formats menu command.</td>
</tr>
<tr>
<td>Get(WindowContentHeight)</td>
<td>A number representing the height, in pixels, of the content area.</td>
</tr>
<tr>
<td>Get(WindowContentWidth)</td>
<td>A number representing the width, in pixels, of the content area.</td>
</tr>
<tr>
<td>Get(WindowDesktopHeight)</td>
<td>A number representing the height, in pixels, of the desktop space.</td>
</tr>
<tr>
<td>Get(WindowDesktopWidth)</td>
<td>A number representing the width, in pixels, of the desktop space.</td>
</tr>
<tr>
<td>Get(WindowHeight)</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)</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)</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)</td>
<td>The name of the current window of the file in which the calculation is defined.</td>
</tr>
<tr>
<td>Get(WindowTop)</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.</td>
</tr>
<tr>
<td>Get(WindowVisible)</td>
<td>A Boolean value representing whether or not the current window is visible.</td>
</tr>
<tr>
<td>Get(WindowWidth)</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)</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.

```plaintext
Go to Layout ["LayoutName"]
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)**

**Purpose**

Returns the name of the authenticated account being used by the current user of the database file.

**Format**

`Get(AccountName)`

**Parameters**

None

**Data type returned**

text
Get functions

**Description**

Use this function for FileMaker authentication. 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 choose 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.
- For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

**Examples**

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

---

**Get(ActiveFieldContents)**

**Purpose**

Returns the contents of the field that has the focus.

**Format**

Get(ActiveFieldContents)

**Parameters**

None

**Data type returned**

text, number, date, time, timestamp, container

**Description**

When the focus is in a repeating field, this function 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** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
Examples
Returns *SomeShop* when the focus 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)

**Purpose**
Returns the name of the field that has the focus.

**Format**
Get(ActiveFieldName)

**Parameters**
None

**Data type returned**
text

**Description**
For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com kb](http://www.filemaker.com/kb).

**Examples**
Returns *Country*, when the focus is in the Country field.

Get(ActiveFieldTableName)

**Purpose**
Returns the name of the table that contains the active field (the field that has the focus).

**Format**
Get(ActiveFieldTableName)

**Parameters**
None

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

If there is no active field, this function returns an empty string.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

**Examples**

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)**

**Purpose**

Returns the object name of the active layout object in the calculation's current window; otherwise, returns an empty string.

**Format**

Get(ActiveLayoutObjectName)

**Parameters**

None

**Data type returned**

text

**Description**

For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

**Examples**

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

---

**Get(ActiveModifierKeys)**

**Purpose**

Returns a number representing the keyboard modifier keys (for example, Control+Shift) that are being pressed.
Get functions

Format
Get(ActiveModifierKeys)

Parameters
None

Data type returned
number

Description
The number returned 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
- Command (Mac OS) = 16

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
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)

Purpose
Returns a number representing the active repetition of a repeating field (the repetition that has the focus).

Format
Get(ActiveRepetitionNumber)

Parameters
None

Data type returned
number
**Description**
The first repetition returns 1. If the current field isn’t a repeating field, this function returns 1.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns 5 when the focus is in the fifth repetition of a repeating field.

---

**Get(ActiveSelectionSize)**

**Purpose**
Returns a number representing how many characters are selected.

**Format**
Get(ActiveSelectionSize)

**Parameters**
None

**Data type returned**
number

**Description**
Returns 0 if there is no selection.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns 4 when 4 characters are selected.

---

**Get(ActiveSelectionStart)**

**Purpose**
Returns a number representing the starting character of the selected text.

**Format**
Get(ActiveSelectionStart)

**Parameters**
None
Data type returned
number

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 5 when the selection starts at character 5.

Get(AllowAbortState)

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

Format
Get(AllowAbortState)

Parameters
None

Data type returned
number

Description
For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(AllowToolbarState)

Purpose
Returns a Boolean value representing whether toolbars are allowed to be visible.
Get functions

Format
Get(AllowToolbarState)

Parameters
None

Data type returned
number

Description
Returns 1 if toolbars are allowed; otherwise, returns 0. The Allow Toolbars script step sets the toolbar state. For more information, see Allow Toolbars script step.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 1 if toolbars are allowed to be visible.

Get(ApplicationLanguage)

Purpose
Returns text representing the current application language.

Format
Get(ApplicationLanguage)

Parameters
None

Data type returned
text

Description
The text that is returned by this function is in the English language. FileMaker Pro supports:
- English
- French
- Italian
- German
- Swedish
- Spanish
Get functions

• Dutch
• Japanese

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns English when the current application language is English.

Get(ApplicationVersion)

Purpose
Returns text representing the FileMaker application and version.

Format
Get(ApplicationVersion)

Parameters
None

Data type returned
text

Description
Returns:
• 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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns Pro 10.0v1 in FileMaker Pro 10.0v1.

Get(CalculationRepetitionNumber)

Purpose
Returns a number representing the repetition of the calculation field that is currently being calculated.
Get functions

Format
Get(CalculationRepetitionNumber)

Parameters
None

Data type returned
number

Description
The first repetition returned is 1. If the current field isn’t a repeating field, the function returns 1.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(CurrentDate)

Purpose
Returns the current date according to the system calendar.

Format
Get (CurrentDate)

Parameters
None

Data type returned
date

Description
The format of the result of this function 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 and time formats in your operating system.

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

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Examples
Returns 2/2/2010 when the system date is set to February 2, 2010.

Get(CurrentHostTimestamp)

Purpose
Returns the host's current date and time (to the nearest second) according to the system clock.

Format
Get(CurrentHostTimestamp)

Parameters
None

Data type returned
timestamp

Description
The format of the value returned is determined by the database file’s settings. You can use your client system's settings in the operating system.

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.
- For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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.

Examples
Returns 1/1/2010 11:30:01 AM when the system clock shows January 1, 2010 11:30:01 AM on the host machine.
Get functions

Get(CurrentTime)

**Purpose**
Returns *CurrentTime* (to the nearest second) according to the system clock.

**Format**
Get(CurrentTime)

**Parameters**
None

**Data type returned**
time

**Description**
The format of the value returned is determined by the operating system settings.

*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). For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns **11:30:00** when the system clock shows 11:30:00.

Get(CurrentTimestamp)

**Purpose**
Returns the current date and time (to the nearest second) according to the system clock.

**Format**
Get(CurrentTimestamp)

**Parameters**
None

**Data type returned**
timestamp

**Description**
The format of the value returned is determined by the operating system settings.

*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). For information on how
functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns 1/1/2010 11:30:00 AM when the system clock shows January 1, 2010 11:30:00.

---

**Get(CustomMenuSetName)**

**Purpose**
Returns the name of the active custom menu set.

**Format**
Get(CustomMenuSetName)

**Parameters**
None

**Data type returned**
text

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

*Note* For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**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)**

**Purpose**
Returns the path to the desktop folder for the current user.

**Format**
Get(DesktopPath)

**Parameters**
None
Data type returned
text

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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(DocumentsPath)

Purpose
Returns the path to the Documents folder for the current user.

Format
Get(DocumentsPath)

Parameters
None

Data type returned
text

Description
In Windows XP, the path format is /Drive:/Documents and Settings/UserName/My Documents/.
In Windows Vista, the path format is /Drive:/Users/UserName/Documents/.
In the Mac OS, the path format is /DriveName/Users/UserName/Documents/.

When running on FileMaker Server, Get(DocumentsPath) returns the location of the Documents folder, which is in the same folder as the server's Backups, Databases, and Scripts folders. The Documents folder is used as a shared location that scripts from different sessions or other processes on the machine can use to import or export files.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
For FileMaker Pro, for a user named John Smith, returns:

/C:/Documents and Settings/JohnSmith/My Documents in Windows XP
Get functions

Get(DocumentsPathListing)

Purpose
Returns a list of all the files and folders in the Documents folder returned by the Get(DocumentsPath) function.

Format
Get(DocumentsPathListing)

Parameters
None

Data type returned
text

Description
Each pathname in the Documents folder is listed separated by a line break. Files and folders are named according to FileMaker Pro naming conventions.

Use Get(DocumentsPathListing) with the Import Records script step and Export Records script step to determine if a file exists in the Documents folder before using the Open File script step to open the file. Get(DocumentsPathListing) ensures that multiple scripts can safely read from and write to the same FileMaker Pro database.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
For FileMaker Server, returns the following pathnames:

In Windows Vista:
/C:/Program Files/FileMaker/FileMaker Server/Data/Documents/lastmonthsales.xls
/C:/Program Files/FileMaker/FileMaker Server/Data/Documents/forecastsales.xls
/C:/Program Files/FileMaker/FileMaker Server/Data/Documents/SAP
/C:/Program Files/FileMaker/FileMaker Server/Data/Documents/SAP/sap001.txt
/C:/Program Files/FileMaker/FileMaker Server/Data/Documents/SAP/sap002.txt

In the Mac OS:
Get functions

/MacintoshHD/Library/FileMaker Server/Data/Documents/lastmonthsales.xls
/MacintoshHD/Library/FileMaker Server/Data/Documents/forecastsales.xls
/MacintoshHD/Library/FileMaker Server/Data/Documents/SAP
/MacintoshHD/Library/FileMaker Server/Data/Documents/SAP/sap001.txt
/MacintoshHD/Library/FileMaker Server/Data/Documents/SAP/sap002.txt

Get(LErrorCaptureState)

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

Format
Get(LErrorCaptureState)

Parameters
None

Data type returned
number

Description
For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 1 if the Set Error capture script step is on.

Get(ExtendedPrivileges)

Purpose
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.

Format
Get(ExtendedPrivileges)

Parameters
None

Data type returned
text
Get functions

Description
Extended privileges are additional access rights assigned to an account’s privilege set. For more information, see About accounts, privilege sets, and extended privileges. 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.
- For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
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)

Purpose
Returns the path to the folder of the currently running copy of FileMaker Pro.

Format
Get(FileMakerPath)

Parameters
None

Data type returned
text

Description
In Windows, the path format is /Drive:/Program Files/FileMaker/FileMaker Pro 10.0/.
In the Mac OS, the path format is /DriveName/Applications/FileMaker Pro 10.0/.
Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns /C:/Program Files/FileMaker/FileMaker Pro 10.0/ in Windows.
Returns /MacintoshHD/Applications/FileMaker Pro 10.0/ in the Mac OS.
Get(FileName)

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

**Format**
Get(FileName)

**Parameters**
None

**Data type returned**
text

**Description**
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.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns Contacts when Contacts is the active file.

Get(FilePath)

**Purpose**
Returns the full path indicating the location of the currently active database file.

**Format**
Get(FilePath)

**Parameters**
None

**Data type returned**
text

**Description**
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.
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** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**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)

**Purpose**

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

**Format**

```
Get(FileSize)
```

**Parameters**

None

**Data type returned**

Number

**Description**

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.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**

- Returns **15000** when the current file size is 15000 bytes.

### Get(FoundCount)

**Purpose**

Returns a number that represents the number of records in the current found set.
Get functions

Get(FoundCount)

Parameters
None

Data type returned
number

Description
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.
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(HighContrastColor)

Purpose
Returns the name of the current high contrast default color scheme if Use High Contrast is selected in the Windows OS Accessibility Options dialog box.

Format
Get(HighContrastColor)

Parameters
None

Data type returned
text

Description
Returns an empty value (null) if Use High Contrast is unavailable, inactive, or if this function is used on the Mac OS.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
Examples

Returns **High Contrast White** when the Windows Vista color scheme is set to High Contrast White.

Get(HighContrastState)

**Purpose**

Returns a Boolean value representing the state of the **Use High Contrast** option in the Accessibility Options dialog box (Windows XP) or the **Choose a High Contrast** color scheme setting in the Ease of Access dialog box (Windows Vista).

**Format**

Get(HighContrastState)

**Parameters**

None

**Data type returned**

number

**Description**

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** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

Get(HostApplicationVersion)

**Purpose**

Returns the version of FileMaker Pro or FileMaker Server running on the computer that is hosting the current database.

**Format**

Get(HostApplicationVersion)

**Parameters**

None

**Data type returned**

text
Description
Displays a value when used with the same or higher version of FileMaker Pro or FileMaker Server software. If the current database is not shared or hosted, this function returns an empty string. Also returns an empty string when used from the host computer itself.

Examples
Returns Pro 10.0v1 when the host computer is running FileMaker Pro 10 version 1.
Returns ProAdvanced 10.0v1 when the host computer is running FileMaker Pro 10 Advanced version 1.
Returns Server 10.0v1 when the host computer is running FileMaker Server 10 version 1.

Get(HostIPAddress)

Purpose
Returns the IP address of the host machine for the current database.

Format
Get(HostIPAddress)

Parameters
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.

If IPv4 and IPv6 addresses are available for remotely hosted files, the address is returned in the most common or default format. This might not be the same format that was used when connecting to the host.

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.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns one of the following when the current database is being hosted:

- IPv4: 14.156.13.121

Note If the host machine has both an IPv4 and IPv6 address, both addresses are returned.
Get(HostName)

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

**Format**
Get(HostName)

**Parameters**
None

**Data type returned**
text

**Description**
On the computer that is hosting the database file:
- Windows XP: Choose Start menu > Control Panel > Performance and Maintenance > System > and then click the Computer Name tab. Full computer name displays the current registered name.
- Windows Vista: Choose Start menu > Control Panel > System and Maintenance > System. Full computer name displays the current registered name.
- 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.
- For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(LastError)

**Purpose**
Returns a number representing the error, if any, in the execution of the most recently executed script step.

**Format**
Get(LastError)
**Parameters**

None

**Data type returned**

number

**Description**

Use this function to detect and control the outcome of errors. See [FileMaker Pro error codes](#).

**Notes**

- Mac OS: In FileMaker Pro, if an error occurs while performing an AppleScript from the Manage Scripts feature, the AppleScript error code will be returned.
- For ODBC imports and Execute SQL script steps, if an error occurs while performing a SQL query, returns FileMaker error 1408. For detailed information about the error, use the `Get(LastODBCError)` function. If there is no information about the error, returns FileMaker error 1409.
- For working with ODBC data sources in the relationships graph, returns FileMaker error 1408.
- Some script triggers allow for the activating command or event to be canceled if the script executed by the script trigger returns a False value. When a command or event is canceled this way, the error code is set to 20.
- When you perform a script that uses this function with control script steps, the control script steps do not clear the last error condition reported by FileMaker Pro.
- For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**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.

---

**Get(LastMessageChoice)**

**Purpose**

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

**Format**

Get(LastMessageChoice)

**Parameters**

None
Data type returned
number

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Get(LastODBCError)

Purpose
Returns a string that shows the error state published by ODBC standards, based on ISO/IEF standards.

Format
Get(LastODBCError)

Parameters
None

Data type returned
text

Description
• For ODBC imports and Execute SQL script steps, returns a detailed, textual ODBC error message.
• For working with ODBC data sources in the relationships graph, returns the readable error string that is generated by the ODBC driver.

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 error codes.
• For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
For ODBC imports and Execute SQL script steps, 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)

**Purpose**
Returns a number based on record access privileges available through the current layout.

**Format**
Get(LayoutAccess)

**Parameters**
None

**Data type returned**
number

**Description**
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

**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.
- For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
- See Editing layouts privileges for more details about limiting access through layouts.

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

Get(LayoutCount)

**Purpose**
Returns the total number of layouts in the database file.
Get functions

Get(LayoutCount)

**Parameters**
None

**Data type returned**
number

**Description**
For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns 3 when the file has three layouts.

Get(LayoutName)

**Purpose**
Returns the name of the layout currently displayed.

**Format**
Get(LayoutName)

**Parameters**
None

**Data type returned**
text

**Description**
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** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns **Product List** when the Product List layout is displayed.
Returns **Customer Invoice** when the Customer Invoice layout is displayed.
Get(LayoutNumber)

Purpose
Returns the number of the layout currently displayed, according to the list in the Manage Layouts dialog box.

Format
Get(LayoutNumber)

Parameters
None

Data type returned
number

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 3 when the current layout is third in the list of layouts in Manage Layouts.

Get(LayoutTableName)

Purpose
Returns the name of the table from which the current layout is displaying records.

Format
Get(LayoutTableName)

Parameters
None

Data type returned
text

Description
If no windows are open, an empty string is returned.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
Examples

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)

Purpose

Returns a number indicating the currently active database file view.

Format

Get(LayoutViewState)

Parameters

None

Data type returned

number

Description

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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Get(MultiUserState)

Purpose

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

Format

Get(MultiUserState)

Parameters

None
**Data type returned**

number

**Description**

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.
- For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

**Examples**

Returns 0 when access is denied to other users.

---

**Get(NetworkProtocol)**

**Purpose**

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

**Format**

Get(NetworkProtocol)

**Parameters**

None

**Data type returned**

text

**Description**

For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
**Examples**

Returns TCP/IP.

---

**Get(PageNumber)**

**Purpose**

Returs a number representing the current page being printed or previewed.

**Format**

Get(PageNumber)

**Parameters**

None

**Data type returned**

number

**Description**

If nothing is being printed or previewed, 0 is returned.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**

Returns 4 when page 4 is being printed or previewed.

---

**Get(PortalRowNumber)**

**Purpose**

Returns the number of the portal row containing the focus.

**Format**

Get(PortalRowNumber)

**Parameters**

None

**Data type returned**

number
Get functions

Description
When no portal row contains the focus, this function 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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 5 when the fifth row of a portal has the focus, or when the focus is in a field in the fifth portal row.

Get(PreferencesPath)

Purpose
Returns the path to the preferences and default options folder for the current user.

Format
Get(PreferencesPath)

Parameters
None

Data type returned
text

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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 functions

Get(PrinterName)

**Purpose**
Returns a string identifying the default printer name.

**Format**
Get(PrinterName)

**Parameters**
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** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com_kb](http://www.filemaker.com_kb).

**Examples**
Returns **HP LaserJet 4, WINSPOOL, LPT1** in Windows.
Returns **24.109.265.43** in the Mac OS.

Get(PrivilegeSetName)

**Purpose**
Returns the name of the privilege set assigned to the account being used by the current user of the database file.

**Format**
Get(PrivilegeSetName)
Get functions

Parameters
None

Data type returned
text

Description
If a user is using the default Admin account and you haven’t modified access privileges for the database file, this function 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.
• If you select the Run script with full access privileges script option, this function returns [Full Access].
• For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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)

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

Format
Get(RecordAccess)

Parameters
None

Data type returned
number

Description
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

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

• 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

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 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.
• For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
• See Editing record access privileges for more details about limiting access to records.

Examples
Returns 1 when the record access is view-only.

Get(RecordID)

Purpose
Returns the unique ID number of the current record.

Format
Get(RecordID)

Parameters
None

Data type returned
number

Description
The number returned is a decimal value (an integer) generated by FileMaker Pro when the record is created. It does not change.
Get functions

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.
• For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
• Get(RecordID) may not return a consistent value for records in ODBC data sources.

Examples
Returns a unique ID for the current record.

Get(RecordModificationCount)

Purpose
Returns the total number of times changes to the current record have been committed.

Format
Get(RecordModificationCount)

Parameters
None

Data type returned
number

Description
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.
• For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
• Get(RecordModificationCount) returns NULL for ODBC data sources.

Examples
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)

**Purpose**
Returns the number of the current record in the current found set.

**Format**
Get(RecordNumber)

**Parameters**
None

**Data type returned**
number

**Description**
The value returned is determined by the relative place of the record in the found set, and it 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.
- For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

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

Get(RecordOpenCount)

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

**Format**
Get(RecordOpenCount)
Get functions

Parameters
None

Data type returned
number

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(RecordOpenState)

Purpose
Returns a number representing the state of the current record.

Format
Get(RecordOpenState)

Parameters
None

Data type returned
number

Description
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.
• For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
Examples
Returns 1 if the current record is a new record that hasn’t been saved.

Get(RequestCount)

Purpose
Returns the total number of find requests defined for the current table.

Format
Get(RequestCount)

Parameters
None

Data type returned
number

Description
If there are multiple windows open in the current database file, then results are returned for only the top-most window of the file in which the calculation is defined.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(RequestOmitState)

Purpose
Returns a Boolean value representing the state of the Omit checkbox in Find mode.

Format
Get(RequestOmitState)

Parameters
None

Data type returned
number
Description

Returns 1 if the Omit checkbox is selected; otherwise, returns 0.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples

Returns 1 when the Omit checkbox is selected in the current find request.

Get(ScreenDepth)

Purpose

Returns the number of bits needed to represent the color or shade of gray of a pixel on the main screen.

Format

Get(ScreenDepth)

Parameters

None

Data type returned

number

Description

A value of 8 represents 256 (equal to $2^8$) colors or shades of gray.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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)

Purpose

Returns the number of pixels displayed vertically on the screen in which the window of the current file is open.
Get functions

Get(ScreenHeight)

Format
Get(ScreenHeight)

Parameters
None

Data type returned
number

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 480 when the screen resolution is set to 640 x 480.

Get(ScreenWidth)

Purpose
Returns the number of pixels displayed horizontally on the screen in which the window of the current file is open.

Format
Get(ScreenWidth)

Parameters
None

Data type returned
number

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.
Examples
Returns 640 when the screen resolution is set to 640 x 480.

Get(ScriptName)

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

Format
Get(ScriptName)

Parameters
None

Data type returned
text

Description
For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(ScriptParameter)

Purpose
Returns the script parameter passed into the current script.

Format
Get(ScriptParameter)

Parameters
None

Data type returned
text
**Description**

Use this function as part of a calculation evaluated within a script.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

**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):

```plaintext
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):

```plaintext
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):

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

**Get(ScriptResult)**

**Purpose**

Returns the script result from a performed subscript.

**Format**

`Get(ScriptResult)`

**Parameters**

None

**Data type returned**

text, number, date, time, timestamp, container

**Description**

Use this function as part of a calculation evaluated within a script. If a subscript doesn’t return a result, then the content of the script result will be empty.
Examples
In the following example, the Find Customers script returns the results of a find request when it is called from the Do Reports script. 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)

Purpose
Returns a value representing the current sort state.

Format
Get(SortState)

Parameters
None

Data type returned
number

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

Notes
• When records are imported from another file to a previously found and sorted set, the records in a sorted set may exist in a semi-sorted state. To include the imported records in the sort order, sort the found set after importing.
• 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.
• For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(StatusAreaState)

Purpose
Returns a number indicating the current status toolbar state.

Format
Get(StatusAreaState)

Parameters
None

Data type returned
number

Description
Returns:
• 0 (zero) if the status toolbar is hidden
• 1 if the status toolbar is visible
• 2 if the status toolbar is visible and locked
• 3 if the status toolbar 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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 1, when the current status toolbar is visible.
Get(SystemDrive)

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

**Format**
Get(SystemDrive)

**Parameters**
None

**Data type returned**
text

**Description**
For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**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)

**Purpose**
Returns a list of the IP addresses of all computers connected to an active NIC (Network Interface Controller) card.

**Format**
Get(SystemIPAddress)

**Parameters**
None

**Data type returned**
text

**Description**
IP addresses are separated by carriage returns.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).
Examples

Suppose a machine has the following active physical interfaces:
• an Ethernet card not connected to a network with an IP address of 10.10.10.10
• a WiFi interface with an IP address of 192.168.1.1
• a VPN connection with an IP address of 172.172.172.172
The function returns:
192.168.1.1
172.172.172.172

Suppose a machine has the following active physical interfaces:
• an Ethernet card not connected to a network with an IP address of 2001::10
• a WiFi interface with an IP address of 3FFE:FFFF:101::230:6EFF:FE04:D9FF/48
• a VPN connection with an IP address of 2001:0DB8:85A3:08D3:1319:8A2E:0370:7334
The function returns:

Get(SystemLanguage)

Purpose
Returns the language currently set on the current system.

Format
Get(SystemLanguage)

Parameters
None

Data type returned
text

Description
The text that is returned is in the English language.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

**Purpose**
Returns a list of the hardware addresses of all NIC (Network Interface Controller) cards connected to the computer.

**Format**
Get(SystemNICAddress)

**Parameters**
None

**Data type returned**
text

**Description**
Values in the list returned by this function are separated by carriage returns. 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** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(SystemPlatform)

**Purpose**
Returns a number indicating the current platform.

**Format**
Get(SystemPlatform)

**Parameters**
None

**Data type returned**
number

**Description**
Returns:
Get functions

-1 if the current platform is PowerPC-based Macs
1 if the current platform is Intel-based Macs
-2 if the platform is Windows XP or Windows Vista

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples

Get(SystemPlatform) returns -2 when the current platform is a Windows platform.
Abs(Get(SystemPlatform)) returns 1 when the current platform is Mac OS X.

Get(SystemVersion)

Purpose
Returns the version of the current operating system.

Format
Get(SystemVersion)

Parameters
None

Data type returned
text

Description
Use this function when running a script.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 6.0 for Windows Vista.
Returns 5.1 for Windows XP (SP 2).
Returns 10.5 for Mac OS X version 10.5.

Get(TemporaryPath)

Purpose
Returns the path to the temporary folder that FileMaker Pro uses for the current user, or the path that FileMaker Server uses on the system.
**Format**

Get(TemporaryPath)

**Parameters**

None

**Description**

The temporary folder name begins with S, followed by a number representing the session of the database engine during which the operation took place. Because your operating system controls the location of temporary files, the exact path returned may be different from the examples shown. The actual path returned also depends on which product (FileMaker Pro or FileMaker Server) is executing the function.

In FileMaker Pro, the temporary folder and any files placed in it are deleted when FileMaker Pro is terminated. In FileMaker Server, each schedule runs in its own session; once the schedule is completed, the session terminates and the temporary folder is deleted.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

**Examples**

In Windows, returns:

```
/%HomeDrive%/Documents and Settings/{user id}/Local Settings/Temp/S<n>
```

or

```
/%UserProfile%/Local Settings/Temp/S<n>
```

where %HomeDrive% is an environment variable that returns the name of your home drive on your hard disk.

%UserProfile% is an environment variable that points to the directory where the profile of the current user is located.

S<n> is the name of the folder in which the temporary files are placed (for example, S1); <n> is a number representing the database engine session during which the operation took place.

In the Mac OS, returns:

```
/DremeName/private/var/tmp/folders/501/TemporaryItems/FileMaker/S<n>
```

where DriveName is the name of your hard disk.

S<n> is the name of the folder in which the temporary files are placed (for example, S1); <n> is a number representing the database engine session during which the operation took place.

---

**Get(TextRulerVisible)**

**Purpose**

Returns a Boolean value representing whether or not the text ruler is visible.

**Format**

Get(TextRulerVisible)
**Get functions**

**Parameters**
None

**Data type returned**
number

**Description**
Returns 1 if the text ruler is displayed; otherwise, returns 0.

*Note* For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns 1 when the text ruler is visible.

---

**Get(TotalRecordCount)**

**Purpose**
Returns the total number of records in the current table.

**Format**

Get(TotalRecordCount)

**Parameters**
None

**Data type returned**
number

**Description**
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.

*Note* For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns 876 when there are 876 records in the current table.
Get(TriggerKeystroke)

**Purpose**

Returns a string containing the characters that activated an *OnObjectKeystroke* or *OnLayoutKeystroke* script trigger. Multiple characters may be returned when the input comes from an input method editor (IME).

**Format**

Get(TriggerKeystroke)

**Parameters**

None

**Data type returned**

text

**Description**

Returns a value when running a script triggered by an *OnObjectKeystroke* or *OnLayoutKeystroke* script trigger or running a script called from the triggered script; otherwise returns an empty string.

**Examples**

The following code displays the text **Processing input...** when a carriage return is entered:

```plaintext
If [ Code ( Get(TriggerKeystroke) ) = 13 ]
    Show Custom Dialog ["Processing input..."]
End If
```

**Note**

For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

Get(TriggerModifierKeys)

**Purpose**

Returns the state of the keyboard modifier keys (for example, Control+Shift) as they were when a script trigger was activated.

**Format**

Get(TriggerModifierKeys)

**Parameters**

None
Data type returned
number

Description
Returns a value only when called from a script activated by a script trigger or from a sub-script called from the triggered script; otherwise returns an empty string.

- See Get(ActiveModifierKeys) for a description of the values assigned to the keyboard modifier keys.
- See the Code function for a list of navigational keys and the codes returned to a script activated by this trigger.

Time might elapse between when the keys that activated a script trigger are pressed and the script asks for information on the modifier keys. Use Get(TriggerKeystroke) and Get(TriggerModifierKeys) to capture the keys that were active when the script trigger was activated. Use Get(ActiveModifierKeys) to capture any current keys being pressed.

Examples
- The following example will only display a custom dialog box when lowercase “a” is entered:

  If [Get(TriggerKeystroke)=“a” and Get(TriggerModifierKeys)=0]
  
  Show Custom Dialog [“You entered \“a\“.”]
  
  End If

- The value 9 is returned when Shift-Option is pressed on a computer running the Mac OS. If the Option and Shift keys are pressed on a Mac when a script is triggered, Get(TriggerModifierKeys) returns 9, regardless of which modifier keys have been pressed between when the trigger was activated and when the script runs.

Note
For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Get(UserCount)

Purpose
Returns the number of clients currently accessing the file.

Format
Get(UserCount)

Parameters
None

Data type returned
number
Get functions

Description
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 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.
• For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(UserName)

Purpose
Returns the name of the FileMaker Pro user, as specified in the General tab of the Preferences dialog box.

Format
Get(UserName)

Parameters
None

Data type returned
text

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

**Purpose**
Returns a Boolean value representing the state of the *Use System Formats* command in the Format menu.

**Format**
Get(UseSystemFormatsState)

**Parameters**
None

**Data type returned**
number

**Description**
Returns 1 if *Use System Formats* is on; otherwise, returns 0.

**Note**
For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns 1 when *Use System Formats* is on.

Get(WindowContentHeight)

**Purpose**
Returns a number representing the height, in pixels, of the FileMaker Pro content area.

**Format**
Get(WindowContentHeight)

**Parameters**
None

**Data type returned**
number

**Description**
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. It does not include the status toolbar if it is currently showing.
Get functions

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 400 in the Mac OS when the current window height is 437 and the status toolbar isn’t showing.

The example below combines Get(WindowContentHeight) with Get(WindowHeight) to determine the size of the title bar and horizontal scroll bar:

Get(WindowHeight) - Get(WindowContentHeight) returns 37 in the Mac OS when the window height is 437 and the status toolbar isn’t showing.

Get(WindowContentWidth)

Purpose
Returns a number representing the width, in pixels, of the FileMaker Pro content area.

Format
Get(WindowContentWidth)

Parameters
None

Data type returned
number

Description
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. The content area is the space inside these controls.

Note For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 400 in the Mac OS when the current window width is 415.

The example below combines Get(WindowContentWidth) with Get(WindowWidth) to determine the size of the vertical scroll bar:

Get(WindowWidth) - Get(WindowContentWidth) returns 15 in the Mac OS when the window width is 415.
Get(WindowDesktopHeight)

**Purpose**
Returns a number representing the height, in pixels, of the desktop space.

**Format**
Get(WindowDesktopHeight)

**Parameters**
None

**Data type returned**
number

**Description**
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. The main monitor is where the menu bar is located.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**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)

**Purpose**
Returns a number representing the width, in pixels, of the desktop space.

**Format**
Get(WindowDesktopWidth)

**Parameters**
None

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

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. The main monitor is where the menu bar is located.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**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)**

**Purpose**

Returns a number representing the height, in pixels, of the window on which the script is acting (not necessarily the foreground window).

**Format**

Get(WindowHeight)

**Parameters**

None

**Data type returned**

number

**Description**

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** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**

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

**Purpose**
Returns a number representing the horizontal distance, in pixels, of the outer edge of the window on which the script is acting (not necessarily the foreground window) relative to the left-most edge of the screen.

**Format**
Get(WindowLeft)

**Parameters**
None

**Data type returned**
number

**Description**
The origin of the reference coordinate system is at the left-most corner below the menu bar. A negative value indicates the portion of the left side of the window that is hidden.

**Note**
For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**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)

**Purpose**
Returns a number representing the mode FileMaker Pro is in at the time the function is evaluated.

**Format**
Get(WindowMode)

**Parameters**
None

**Data type returned**
number

**Description**
Returns:
Get functions

- 0 for Browse mode
- 1 for Find mode
- 2 for Preview mode
- 3 if printing is in progress
- 4 (FileMaker Pro Advanced) if evaluating the function from the Data Viewer and the current window is in Layout mode

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** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**

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

Get(WindowName)

**Purpose**

Returns the name of the window on which the script is acting (not necessarily the foreground window).

**Format**

Get (WindowName)

**Parameters**

None

**Data type returned**

text

**Description**

Returns an empty string if there is no window.

**Notes**

- You can set the window name with the [Set Window Title script step](http://www.filemaker.com/kb).
- For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**

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)

**Purpose**
Returns a number representing the vertical distance, in pixels, of the outer edge of the window on which the script is acting (not necessarily the foreground window) relative to the bottom edge of the menu bar.

**Format**
Get(WindowTop)

**Parameters**
None

**Data type returned**
number

**Description**
The origin of the reference coordinate system is at the left-most corner below the menu bar. A negative value indicates the portion of the top part of the window that is hidden behind the menu bar.

**Note** For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at [www.filemaker.com/kb](http://www.filemaker.com/kb).

**Examples**
Returns 52 when the outer edge of the active window is 52 pixels from the menu bar.
Returns 0 when the outer edge of the active window just touches the menu bar.

Get(WindowVisible)

**Purpose**
Returns a number representing whether or not the current window is visible.

**Format**
Get(WindowVisible)

**Parameters**
None

**Data type returned**
number
Get functions

Description
The current window is the window on which the script is acting (not necessarily the foreground window). Returns 1 if the window is visible. Returns 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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

Examples
Returns 1 when the current window is physically visible.
Returns 0 when the current window has been hidden using the Hide Window command in FileMaker Pro.

Get(WindowWidth)

Purpose
Returns a number representing the width, in pixels, of the window on which the script is acting (not necessarily the foreground window).

Format
Get(WindowWidth)

Parameters
None

Data type returned
number

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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

Get(WindowZoomLevel)

Purpose
Returns the zoom percentage of the current window.
Format
Get(WindowZoomLevel)

Parameters
None

Data type returned
text

Description
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 For information on how functions evaluate differently on the host versus the client, search the FileMaker Knowledge Base available at www.filemaker.com/kb.

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.
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><strong>Case</strong></td>
<td>One of several possible results based on a series of tests.</td>
</tr>
<tr>
<td><strong>Choose</strong></td>
<td>One result value, according to the integer value of a specified test.</td>
</tr>
<tr>
<td><strong>Evaluate</strong></td>
<td>Evaluates the specified expression as a calculation.</td>
</tr>
<tr>
<td><strong>EvaluationError</strong></td>
<td>An error code, if any, from the specified expression.</td>
</tr>
<tr>
<td><strong>GetAsBoolean</strong></td>
<td>1 if data converts to a non-zero numeric value; 0 if the specified data has a value of 0 or is empty.</td>
</tr>
<tr>
<td><strong>GetField</strong></td>
<td>The contents of the referenced field.</td>
</tr>
<tr>
<td><strong>GetFieldName</strong></td>
<td>The fully qualified name of a field reference.</td>
</tr>
<tr>
<td><strong>GetLayoutObjectAttribute</strong></td>
<td>The requested layout object attributes from the calculation's active window.</td>
</tr>
<tr>
<td><strong>GetNthRecord</strong></td>
<td>The contents of the referenced field from the requested record number.</td>
</tr>
<tr>
<td><strong>If</strong></td>
<td>One of two possible results depending on the value of the specified test.</td>
</tr>
<tr>
<td><strong>IsEmpty</strong></td>
<td>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 0.</td>
</tr>
<tr>
<td><strong>IsValid</strong></td>
<td>0 when a record contains an invalid value because of a field type mismatch (text in a date field, for example).</td>
</tr>
<tr>
<td><strong>IsValidExpression</strong></td>
<td>1 if the syntax of the specified expression is correct.</td>
</tr>
<tr>
<td><strong>Let</strong></td>
<td>Sets variable to the result of value for the duration of the specified expression.</td>
</tr>
<tr>
<td><strong>Lookup</strong></td>
<td>The value specified in the sourceField parameter using the relationships graph.</td>
</tr>
<tr>
<td><strong>LookupNext</strong></td>
<td>The value specified in the sourceField parameter using the relationships graph.</td>
</tr>
<tr>
<td><strong>Self</strong></td>
<td>The content of the object in which the calculation is defined; otherwise, returns an empty string.</td>
</tr>
</tbody>
</table>

Case

**Purpose**

Returns one of several possible results based on a series of tests.
Logical functions

**Format**

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

**Parameters**

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

Parameters in curly braces {} are optional.

**Data type returned**

text, number, date, time, timestamp, container

**Description**

Case 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, Case 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**

**Purpose**

Returns one result value, according to the integer value of test.

**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

FileMaker Pro evaluates test to obtain an index number, which is used to choose the corresponding ordinal result.

Because Choose 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.

Examples

Choose(Rating; "Not Applicable"; "Good"; "Fair"; "Poor")

Rating is a number field that is empty or holds a value. If Rating is empty or 0, the Choose function returns Not Applicable. 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. If Rating contains a value that does not map to one of the result parameters, the Choose function returns nothing.

Evaluate

Purpose

Evaluates expression as a calculation.

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

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**

**Purpose**

Returns an error code, if any, from expression.
**Logical functions**

**Format**

`EvaluationError(expression)`

**Parameters**

*expression* - *any calculation expression*

**Data type returned**

*number*

**Description**

There are two types of errors: syntax and runtime. 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 [FileMaker Pro error codes](#) 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**

**Purpose**

Returns 1 if *data* converts to a non-zero numeric value or if a container field contains data; returns 0 if *data* has a numeric value of 0 or is empty.

**Format**

`GetAsBoolean(data)`

**Parameters**

*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 a Boolean value.
**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**

**Purpose**

Returns the contents of fieldName.

**Format**

GetField(fieldName)

**Parameters**

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

**Important** See [Design functions](#) for information about literal text parameters.

**Data type returned**

text, number, date, time, timestamp, container

**Description**

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

**Examples**

Suppose you have the 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.

Suppose you have the 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.
GetFieldName

**Purpose**
Returns the fully qualified name of a field reference.

**Format**
GetFieldName(fieldName)

**Parameters**
fieldName - any field object or evaluation of a text expression that refers to a field’s name

**Data type returned**
text

**Description**
Use this function to get the fully qualified name of fieldName (tableName::fileName).

**Note** 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.

**Examples**
GetFieldName(Self) returns Personnel::Name from the Name field in the Personnel table.
GetFieldName(x) returns the name of a field reference passed into a custom function as parameter x.
GetFieldName(Evaluate(<fieldName>)) returns the name of a field based on the data stored in <fieldName>.
GetFieldName(Evaluate(Get(ActiveFieldName))) returns the fully qualified name of the field that has the focus when executed.

GetLayoutObjectAttribute

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

**Format**
GetLayoutObjectAttribute(objectName;attributeName{;repetitionNumber;portalRowNumber})

**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).
portalRowNumber - the number of the row in the portal.
Parameters in curly braces {} are optional.

**Data type returned**
text

**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 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 field 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 field 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.

**Notes**

- If objects are set to auto-resize, attributes returned are based on the resized bounds of the object in its current state.
- If objects are located above the status toolbar, negative coordinate values are returned.
- The hasFocus, containsFocus, source, and content attributes behave differently in Instant Web Publishing. For more information, in FileMaker_Pro, choose Help menu > Product Documentation > Instant Web Publishing Guide, or find this guide in the folder where FileMaker Pro is installed.
- When repetitionNumber or portalRowNumber is 0, the function behaves as if the parameter was not specified. For portalRowNumber, the function returns data from the first portal row. For repetitionNumber, the function acts on the first repetition (for returning content or source) or acts on the entire field as a whole (for returning bounds). Both parameters are necessary because you must be able to reference a particular field repetition within a particular portal row.

**Examples**

GetLayoutObjectAttribute("CancelButton","objectType") returns group (if the button does not have an action or script associated with it) or returns button group (if the button has an action or script associated with it).

GetLayoutObjectAttribute("CancelButton","bounds") returns 138 24 391 38 0.

---

**GetNthRecord**

**Purpose**

Returns the contents of fieldName from the provided recordNumber.

**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
Logical functions

Description

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

Purpose

Returns one of two possible results (result1 or result2) depending on the value of test.

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

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, If 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

Purpose
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).

Format
IsEmpty(field)

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

Data type returned
number

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

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

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

IsEmpty(“text”) returns 0.

IsValid

Purpose
Returns 0 (False) if the data is invalid and 1 (True) if the data is valid.

Format
IsValid(field)
Parameters
field - any field name

Data type returned
number

Description
Returns 0 (False) if:
• A record contains an invalid value because of a field 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

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

Format
IsValidExpression(expression)

Parameters
expression - any calculation expression

Data type returned
number

Examples
IsValidExpression(calculationField) returns 1 (true) if calculationField contains total + 1.
isValidExpression(calculationField) returns 0 (false) if calculationField contains abs(-1 with no closing parenthesis.

Let

Purpose
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).

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

Parameters
var - any variable name, local variable name, or global variable name (see About naming fields 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
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.
Logical functions

Let\([x=5;\text{ squared}=x^2;\text{ cubed} = \text{ squared} \times x]\); 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($counter[50]=120;\text{ }$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

Purpose
Returns the value specified in sourceField using the relationships in the relationships graph. The result of the optional failExpression will be returned if the lookup fails.

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
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.

Note Lookup returns ? when the related table is an ODBC data source.
Examples
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

**Purpose**

Returns the value specified in `sourceField` using the relationships in the relationships graph. If the lookup fails, the value from `sourceField` in the lower or higher matching record will be returned, as specified by `lower/higherFlag`.

**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 no related record is found

**Data type returned**

- text, number, date, time, timestamp, container

**Description**

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.
Note LookupNext returns ? when the related table is an ODBC data source.

Examples
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 
CompanyName = LookupNext(Company::CompanyName;Higher) defined in the People table 
will return Apple, FileMaker, Motorola, and Cisco for records 1 to 4.

Self

Purpose
Returns the content of the object in which the calculation is defined.

Format
Self

Parameters
None

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

Description
The Self function provides a way for a calculation to reference the object with which it is associated without having to explicitly reference the object.
Use `Self` to create a single calculation formula that can be applied to different objects. The `Self` function is helpful for conditional formatting calculations and tooltip calculations because it returns the content of the layout object when that object has a value. You can also use the `Self` function in field definition calculations (including auto-enter and validation calculations) to return the value of the corresponding field.

**Examples**

`self > 10` returns `1` (True) when applied to a layout field object whose value is greater than 10.
Number functions

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</td>
<td>The absolute value (a positive number) of a number.</td>
</tr>
<tr>
<td>Ceiling</td>
<td>A number rounded up to the next integer.</td>
</tr>
<tr>
<td>Combination</td>
<td>The number of ways to uniquely choose a specified number of items from a set of specified size.</td>
</tr>
<tr>
<td>Div</td>
<td>An integer of the specified number divided by the divisor.</td>
</tr>
<tr>
<td>Exp</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</td>
<td>The factorial of a specified number stopping at 1, or at a specified number factorial.</td>
</tr>
<tr>
<td>Floor</td>
<td>A number rounded down to the next lower integer.</td>
</tr>
<tr>
<td>Int</td>
<td>The whole number (integer) part of the value you specify, without rounding.</td>
</tr>
<tr>
<td>Lg</td>
<td>The base 2 logarithm of the specified number, which can be any positive value.</td>
</tr>
<tr>
<td>Ln</td>
<td>The base-e (natural) logarithm of the specified number.</td>
</tr>
<tr>
<td>Log</td>
<td>The common logarithm (base 10) of the specified number, which can be any positive value.</td>
</tr>
<tr>
<td>Mod</td>
<td>The remainder after a specified number is divided by divisor.</td>
</tr>
<tr>
<td>Random</td>
<td>A random number between zero and one.</td>
</tr>
<tr>
<td>Round</td>
<td>A number rounded off to the specified precision (number of decimal places).</td>
</tr>
<tr>
<td>SetPrecision</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</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>Sqr</td>
<td>The square root of a number.</td>
</tr>
<tr>
<td>Truncate</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**

**Purpose**
Returns the absolute value of *number*.

**Format**
Abs(number)

**Parameters**

number - any numeric expression or field containing a numeric expression
**Number functions**

**Data type returned**
number, time

**Description**
The absolute value of a number is positive. 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**

**Purpose**
Returns number rounded up to the next integer.

**Format**
Ceiling(number)

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

**Data type returned**
number

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

**Combination**

**Purpose**
Returns the number of ways to uniquely choose numberOfChoices items from a set of size setSize.

**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**

This function is useful in statistics, combinatorics, and polynomial expansions. The values returned by this function are referred to as combination coefficients. They form Pascal’s triangle.

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

**Examples**

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 * 12 * Combination(4;2) * Combination(4;3)) / 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**

**Purpose**

Returns the next lowest integer value after dividing number by divisor. Equivalent to Floor(number/divisor).

**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

**Examples**

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

Purpose
Returns the value of the constant e (the base of the natural logarithm, equal to 2.7182818) raised to the power of number.

Format
Exp(number)

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

Data type returned
number

Description
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

Purpose
Returns the factorial of number stopping at 1 or stopping at the optional numberOfFactors.

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
**Number functions**

**Description**
This function is 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**
$\text{Factorial}(3)$ returns 6, which = $3 \times 2 \times 1$.
$\text{Factorial}(10;3)$ returns 720, which = $10 \times 9 \times 8$.

**Floor**

**Purpose**
Returns number rounded down to the next lower integer.

**Format**
Floor(number)

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

**Data type returned**
number

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

**Int**

**Purpose**
Drops digits to the right of the decimal point and returns the integer part of number without rounding.

**Format**
Int(number)

**Parameters**
number - any numeric expression or field containing a numeric expression
Number functions

Data type returned
number

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

Lg

Purpose
Returns the base-2 logarithm of number.

Format
Lg(number)

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

Data type returned
number

Description
Number can be any positive value. Negative values return an error. For 0, the Lg function returns nothing because this value is out of the acceptable range.

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

Examples
Lg(1) = 0
Lg(2) = 1
Lg(32) = 5

Ln

Purpose
Returns the base-e (natural) logarithm of number.
**Format**

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

**Parameters**

number - any numeric expression or field containing a numeric expression

**Data type returned**

number

**Description**

\( \text{Number} \) can be any positive value. Negative values return an error. For 0, the \( \text{Ln} \) function returns nothing because this value is out of the acceptable range. The \( \text{Exp} \) function is the inverse of the \( \text{Ln} \) function.

**Examples**

\( \text{Ln}(2.7182818) \) returns \( .99999998... \).

\( \text{Ln}(<\text{Exp}(5)> \) returns 5.

---

**Log**

**Purpose**

Returns the base-10 (common) logarithm of number.

**Format**

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

**Parameters**

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

**Data type returned**

number

**Description**

\( \text{Number} \) can be any positive value. Negative values return an error. For 0, the \( \text{Log} \) function returns nothing because this value is out of the acceptable range.

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

**Examples**

\( \text{Log}(1) \) returns 0.

\( \text{Log}(100) \) returns 2.
Mod

**Purpose**
Returns the remainder after `number` is divided by `divisor`.

**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**
Use the `Mod` function to test whether a number is even or odd by specifying a divisor of 2. If the result is zero, then the number is even; otherwise, it’s odd. The result has the same sign as `divisor`.

```
Mod = number - (Div(number;divisor) * 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

**Purpose**
Returns a random number between zero and one.

**Format**
```
Random
```

**Parameters**
- None
Data type returned
number

Description
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

Examples
NumDice + NumSides * Random

Round

Purpose
Returns number rounded off to the specified precision (number of decimal places).

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
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

**Purpose**
Computes any math functions contained within `expression` that support extended precision to `precision` decimal places (from 16 up to 400).

**Format**
```
SetPrecision(expression;precision)
```

**Parameters**
- `expression` - any numeric expression
- `precision` - any number or numeric expression

**Data type returned**
number

**Description**
All functions except 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 0.555555555555555555555555555555556.
- `SetPrecision(If(field1>5;Exp(50);Average(5/9;1/7;5/7));25)` returns either 51847055285872464087.4533229334853848274691006 if `field1 > 5`, or 0.4708994708994708994708999 if `field1 <= 5`.

Sign

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

**Format**
```
Sign(number)
```

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

**Data type returned**
number
**Examples**

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

### Sqrt

**Purpose**

Calculates the square root of `number`.

**Format**

`Sqrt(number)`

**Parameters**

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

**Data type returned**

`number`

**Description**

Use this function to calculate `Sqrt`.

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

**Examples**

- `Sqrt(4)` returns **2**.
- `Sqrt(SquareFeet)` returns **6** if the `SquareFeet` number field contains **36**.

### Truncate

**Purpose**

Returns `number` truncated to the specified precision (number of decimal places).

**Format**

`Truncate(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**

This function doesn’t evaluate digits beyond the specified precision. Use the Round function to round up or down to the required precision.

**Examples**

Truncate(123.456;2) returns **123.45**.
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**.
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</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</td>
<td>The contents of the specified repetition of a repeating field.</td>
</tr>
<tr>
<td>Last</td>
<td>The last valid, non-blank value in the specified field.</td>
</tr>
</tbody>
</table>

Extend

**Purpose**
Allows a value in non-repeatingField to be used with every repetition in a repeating field.

**Format**
Extend(non-repeatingField)

**Parameters**
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**
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

**Purpose**
Returns the contents of the repeating field specified by \texttt{number}.

**Format**
\texttt{GetRepetition(repeatingField;number)}

**Parameters**
- \texttt{repeatingField} - any repeating field, or an expression that returns a reference to a repeating field
- \texttt{number} - the field repetition number

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

**Examples**
ParcelBids is a field defined to repeat with ten values and contains the values 2500, 1200, and 1500.
\texttt{GetRepetition(ParcelBids;2)} returns 1200.
\texttt{GetRepetition(if(IsEmpty(ParcelBids) \neq \text{true}, \; ParcelBids, \; HouseBids);2)} returns 1200.
\texttt{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, \texttt{ParcelBids[2]} returns 1200. See Getting the contents of a repetition in a repeating field.

---

Last

**Purpose**
Returns the last valid, non-blank value in \texttt{field}.

**Format**
\texttt{Last(field)}

**Parameters**
- \texttt{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

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.
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 alternative 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 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</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**

**Purpose**

Returns the value of `summaryField` for the current range of records when the database file is sorted by `breakField`.

**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 break field parameters.

`GetSummary` must be set up in the same table as the break field.

**Data type returned**

number, date, time, timestamp

**Description**

This function 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 `GetSummary` 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. For more information, see Summarizing data in portals.

**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.
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 quotation marks)
- 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>Char</strong></td>
<td>Returns the characters for the Unicode code points in the number.</td>
</tr>
<tr>
<td><strong>Code</strong></td>
<td>Returns the Unicode code points for the characters in the text. If zero characters are in the text, returns 0.</td>
</tr>
<tr>
<td><strong>Exact</strong></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></td>
<td>Only the specified characters, in the order that they were originally entered in the text.</td>
</tr>
<tr>
<td><strong>FilterValues</strong></td>
<td>Only the specified values, in the order that they were originally entered in the text.</td>
</tr>
<tr>
<td><strong>GetAsCSS</strong></td>
<td>The specified text, converted to the CSS (Cascading Style Sheets) format.</td>
</tr>
<tr>
<td><strong>GetAsDate</strong></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></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></td>
<td>The specified text, converted to the SVG (Scalable Vector Graphics) format.</td>
</tr>
<tr>
<td><strong>GetAsText</strong></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></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></td>
<td>The specified data as field type timestamp, for use with formulas involving timestamps.</td>
</tr>
<tr>
<td><strong>GetAsURLEncoded</strong></td>
<td>The specified text, converted with URL (Uniform Resource Locators) encoding.</td>
</tr>
<tr>
<td><strong>GetValue</strong></td>
<td>A specific value from a list of values.</td>
</tr>
<tr>
<td><strong>Hiragana</strong></td>
<td>Hiragana converted from Katakana (Hankaku and Zenkaku).</td>
</tr>
<tr>
<td><strong>KanaHankaku</strong></td>
<td>Hankaku Katakana converted from Zenkaku Katakana.</td>
</tr>
<tr>
<td><strong>KanaZenkaku</strong></td>
<td>Zenkaku Katakana converted from Hankaku Katakana.</td>
</tr>
<tr>
<td><strong>KanjiNumeral</strong></td>
<td>Kanji numerals converted from Arabic numerals.</td>
</tr>
<tr>
<td><strong>Katakana</strong></td>
<td>Zenkaku Katakana converted from Hiragana.</td>
</tr>
<tr>
<td><strong>Left</strong></td>
<td>The specified number of characters in the text, counting from the left.</td>
</tr>
<tr>
<td><strong>LeftValues</strong></td>
<td>The specified number of values in the text, counting from the left.</td>
</tr>
<tr>
<td><strong>LeftWords</strong></td>
<td>The specified number of words in the text, counting from the left.</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>The number of characters in the specified text, including all spaces, numbers, and special characters.</td>
</tr>
<tr>
<td>This function</td>
<td>Returns</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Lower</strong></td>
<td>All letters in the specified text as lowercase.</td>
</tr>
<tr>
<td><strong>Middle</strong></td>
<td>The specified number of characters in the text, starting at a specified character position.</td>
</tr>
<tr>
<td><strong>MiddleValues</strong></td>
<td>The specified number of values in the text, starting with a specified value.</td>
</tr>
<tr>
<td><strong>MiddleWords</strong></td>
<td>The specified number of words in the text, starting with a specified word.</td>
</tr>
<tr>
<td><strong>NumToJText</strong></td>
<td>Roman numbers converted from Japanese text.</td>
</tr>
<tr>
<td><strong>PatternCount</strong></td>
<td>The number of occurrences of a search string in the specified text.</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>The specified occurrence of a search string, starting from a specified position.</td>
</tr>
<tr>
<td><strong>Proper</strong></td>
<td>The first letter of each word in the specified text as uppercase, and all other letters as lowercase.</td>
</tr>
<tr>
<td><strong>Quote</strong></td>
<td>The specified text surrounded by quotation marks (“ ”).</td>
</tr>
<tr>
<td><strong>Replace</strong></td>
<td>A new string of characters consisting of the specified text as modified by the specified replacement text.</td>
</tr>
<tr>
<td><strong>Right</strong></td>
<td>The specified number of characters in the text, counting from the right.</td>
</tr>
<tr>
<td><strong>RightValues</strong></td>
<td>The specified number of values in the text, counting from the right.</td>
</tr>
<tr>
<td><strong>RightWords</strong></td>
<td>The specified number of words in the text, counting from the right.</td>
</tr>
<tr>
<td><strong>RomanHankaku</strong></td>
<td>Hankaku (alphanumeric &amp; symbols) converted from Zenkaku (alphanumeric &amp; symbols).</td>
</tr>
<tr>
<td><strong>RomanZenkaku</strong></td>
<td>Zenkaku (alphanumeric &amp; symbols) converted from Hankaku (alphanumeric &amp; symbols).</td>
</tr>
<tr>
<td><strong>SerialIncrement</strong></td>
<td>The combined text and numbers in a specified value, with the numbers incremented by the specified amount.</td>
</tr>
<tr>
<td><strong>Substitute</strong></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><strong>Trim</strong></td>
<td>Text stripped of all leading and trailing spaces.</td>
</tr>
<tr>
<td><strong>TrimAll</strong></td>
<td>Text with full width spaces between non-Roman and Roman characters removed.</td>
</tr>
<tr>
<td><strong>Upper</strong></td>
<td>All letters in the specified text as uppercase.</td>
</tr>
<tr>
<td><strong>ValueCount</strong></td>
<td>A count of the total number of values in the specified text.</td>
</tr>
<tr>
<td><strong>WordCount</strong></td>
<td>A count of the total number of words in the specified text.</td>
</tr>
</tbody>
</table>

**Char**

- **Purpose**
  Returns the characters for the Unicode code points in the number.

- **Format**
  
  `Char(number)`

- **Parameters**
  
  `number` - a number representing one or more Unicode code points
Data type returned
text

Description
Each group of five digits in the number is treated as a Unicode code point, and the character for each five-digit group is returned in the text.

If the number is 0, the function returns an empty string.
If the number is between 1 and 99,999, the function returns a single character.
If the number contains more than five digits, the function returns the string of characters represented by those code points.

Note Some Unicode characters can be represented by a single code point or multiple code points. For example, the character ä can be represented by the letter a plus ¨ (dieresis) or by the single character ä. The single code point version of this kind of character is called a precomposed character or a composite character.

Examples
Char(0) returns an empty string ("").
Char(97) returns a.
Char(98) returns b.
Char(9800097) returns ab.
Char(228) returns ä.
Char(77600097) returns ä. In this case the number represents two Unicode characters: the letter a and the dieresis character. When these two characters appear together in a string they are displayed as a single character.

Code

Purpose
Returns the Unicode code points for the characters in the text.

Format
Code(text)

Parameters
text - one or more characters

Data type returned
number

Description
Returns the Unicode code points for the characters in the text. If zero characters are in the text, returns 0.
If one character is in the text, the function returns the code point for that character. If the text contains multiple characters, the Unicode code point for each character is returned as a group of five digits where the code point for the first character is represented by the low five digits, the code point for the second character in the next higher (to the left) five digits, and so forth.

When converting a composite character such as ä (an a plus a dieresis character), the function returns the Unicode code point for the composite character.

The following table shows how navigational characters are reported to a script activated by this trigger:

<table>
<thead>
<tr>
<th>Key Pressed</th>
<th>Is reported as</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>backspace</td>
<td>8</td>
<td>Corresponds to Unicode/ASCII code for BS (backspace)</td>
</tr>
<tr>
<td>tab</td>
<td>9</td>
<td>Corresponds to Unicode/ASCII code for HT (horizontal tab)</td>
</tr>
<tr>
<td>shift-tab</td>
<td>9</td>
<td>The shift can be detected using the value returned from the Get(TriggerModifierKeys) function</td>
</tr>
<tr>
<td>enter</td>
<td>10</td>
<td>Corresponds to Unicode/ASCII code for LF (linefeed)</td>
</tr>
<tr>
<td>return</td>
<td>13</td>
<td>Corresponds to Unicode/ASCII code for CR (carriage return)</td>
</tr>
<tr>
<td>escape</td>
<td>27</td>
<td>Corresponds to Unicode/ASCII code for ESC (escape)</td>
</tr>
<tr>
<td>left arrow</td>
<td>28</td>
<td>Corresponds to Unicode/ASCII code for FS (file separator)</td>
</tr>
<tr>
<td>up arrow</td>
<td>29</td>
<td>Corresponds to Unicode/ASCII code for GS (group separator)</td>
</tr>
<tr>
<td>right arrow</td>
<td>30</td>
<td>Corresponds to Unicode/ASCII code for RS (record separator)</td>
</tr>
<tr>
<td>down arrow</td>
<td>31</td>
<td>Corresponds to Unicode/ASCII code for US (unit separator)</td>
</tr>
<tr>
<td>space</td>
<td>32</td>
<td>Corresponds to Unicode/ASCII code for Space</td>
</tr>
<tr>
<td>forward delete</td>
<td>127</td>
<td>Corresponds to Unicode/ASCII code for Delete</td>
</tr>
</tbody>
</table>

**Note** If there are too many characters to be represented in the FileMaker number field type, the function returns a NaN (Not a Number) value.

**Examples**

Code (“”) returns 0.
Code (“a”) returns 97.
Code (“b”) returns 98.
Code (“ab”) returns 9800097.
Code (“ä”) returns 228.

**Exact**

**Purpose**

Compares the contents of any two fields. If the fields match, the result is 1 (True); otherwise, the result is 0 (False).
**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**

For text to match exactly, the uppercase and lowercase usage must be the same. For container fields, the data must be stored in the same manner (either embedded, or stored by 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**

**Purpose**

Returns from textToFilter only those characters specified in filterText, in the order they were originally entered in textToFilter.

**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**
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;7;4))

If `theField` contains **Work: 408.555.1212** this calculation returns **(408)555-1212**.

**FilterValues**

**Purpose**
Returns a text result containing only the values that were provided in `filterValues`, in the order they were originally entered in `textToFilter`.

**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 [Design functions](#) for information about literal text parameters.

**Data type returned**

text

**Description**
If `filterValues` doesn’t have any values, 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.
The `FilterValues` function is not case-sensitive.

**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**

**Purpose**

Returns text converted to the CSS (Cascading Style Sheets) format.

**Format**

GetAsCSS(text)

**Parameters**

text - any text expression or text field

**Data type returned**

text

**Description**

CSS format is an internet text format similar to HTML. CSS supports more text formats than HTML, so CSS will represent what you have typed more accurately.

**Note** The `GetAsCSS` function does not return formats that are set in the Conditional Formatting dialog box.

**Examples**

`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

**Purpose**
Returns dates in text as field type date, for use in formulas involving dates or date functions.

**Format**
GetAsDate(text)

**Parameters**
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**
Use the GetAsDate or Date function to enter a date constant into a formula. The format of text date must be the same as the date format on the system where the file was created.

**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 [Conversion of dates with two-digit years](#).

---

**Examples**
GetAsDate("03/03/2010") returns 3/3/2010. You can perform date operations on this result.

---

GetAsNumber

**Purpose**
Returns only the numbers in text, as field type number, for use with formulas involving numbers or numeric functions.

**Format**
GetAsNumber(text)

**Parameters**
text - any text expression or text field containing numbers

**Data type returned**
number
Description
This 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

Purpose
Returns text converted to the SVG (Scalable Vector Graphics) format.

Format
GetAsSVG(text)

Parameters
text - any text expression or text field

Data type returned
text

Description
SVG is 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.

Note The GetAsSVG function does not return formats that are set in the Conditional Formatting dialog box.

Examples
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>
</StyleList>
<Data>
GetAsText

**Purpose**
Returns data as field type text, for use with formulas involving text or text functions.

**Format**
GetAsText(data)

**Parameters**
data - any number, date, time or timestamp expression, or a field containing a number, date, time, timestamp, or container

**Data type returned**
text

**Description**
The data returned can be a field 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

**Purpose**
Returns times or timestamps in text as field type time, for use with formulas involving the Time or Timestamp functions.

**Format**
GetAsTime(text)

**Parameters**
text - any text expression or text field containing a time
Data type returned

**Time**

**Description**

Use the `GetAsTime` or the `Time` function to enter a time constant into a formula. The format of the supplied time must be the same as the time format on the system where the file was created.

**Examples**

```plaintext
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**

**Purpose**

Returns text as field type timestamp, for use with formulas involving timestamps.

**Format**

`GetAsTimestamp(text)`

**Parameters**

text - any text expression, or text, number, date, or time field

**Data type returned**

timestamp

**Description**

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**

```plaintext
GetAsTimestamp(50000) returns 1/1/0001 1:53:20 PM.
```

---

**GetAsURLEncoded**

**Purpose**

Returns text as URL (Uniform Resource Locator) encoding, for use as a URL.
Format
GetAsURLEncoded(text)

Parameters
text - any text expression or text field

Data type returned
text

Description
This function 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:
www.w3.org

Examples
GetAsURLEncoded("Hello") returns Hello.
GetAsURLEncoded("San Francisco") returns San%20Francisco.
GetAsURLEncoded("français") returns fran%c3%a7ais.

GetValue

Purpose
Returns the requested value given by valueNumber from listOfValues.

Format
GetValue(listOfValues;valueNumber)

Parameters
listOfValues - a list of carriage return-delimited values
valueNumber - the value to return from the list

Data type returned
text

Description
This function is 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.

**Examples**

`GetValue("London¶Paris¶Hong Kong";2)` returns

Paris

---

**Hiragana**

**Purpose**

Converts Katakana (Hankaku and Zenkaku) in text to Hiragana.

**Format**

`Hiragana(text)`

**Parameters**

`text` - any text expression or text field

**Data type returned**

text

**Examples**

`Hiragana("アイウエオ")` returns あいうえお

---

**KanaHankaku**

**Purpose**

Converts Zenkaku Katakana to Hankaku Katakana.

**Format**

`KanaHankaku(text)`

**Parameters**

`text` - any text expression or text field

**Data type returned**

text
Examples
KanaHankaku(“データベース”) returns データベース

KanaZenkaku

Purpose
Converts Hankaku Katakana to Zenkaku Katakana.

Format
KanaZenkaku(text)

Parameters
text - any text expression or text field

Data type returned
text

Examples
KanaZenkaku(“データベース”) returns データベース

KanjiNumeral

Purpose
Converts Arabic numerals to Kanji numeral.

Format
KanjiNumeral(text)

Parameters
text - any text expression or text field

Data type returned
text

Examples
KanjiNumeral(123) returns 一二三
KanjiNumeral(“富士見台二の三の二五”) returns 富士見台二の三の二五
Katakana

**Purpose**
Converts from Hiragana to Zenkaku Katakana.

**Format**
Katakana(text)

**Parameters**
text - any text expression or text field

**Data type returned**
text

**Examples**
Katakana("あいうえお") returns アイウエオ

Left

**Purpose**
Returns numberOfCharacters in text, counting from the 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

**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

**Purpose**
Returns a text result containing \texttt{numberOfValues} from the list of values in \texttt{text}, counting from the left.

**Format**
\texttt{LeftValues(text;numberOfValues)}

**Parameters**
text - any text expression or text field
numberOfValues - any numeric expression or field containing a number

**Important** See Design functions for information about literal text parameters.

**Data type returned**
text

**Description**
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**
\texttt{LeftValues("Plaid¶Canvas¶Suitcase";2)} returns
\texttt{Plaid}
\texttt{Canvas}
\texttt{LeftValues(list;1)} returns
\texttt{Sophie}
when the text being evaluated contains
\begin{itemize}
  \item Sophie
  \item Bill
\end{itemize}

LeftWords

**Purpose**
Returns a text result containing \texttt{numberOfWords} in \texttt{text}, counting from the left.
**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

**Examples**

LeftWords("Plaid Canvas Suitcase";2) returns Plaid Canvas.

LeftWords(Name;1) returns Sophie, when the Name field contains Sophie Tang.

**Note** The ampersand (&) and hyphen (-) characters identify the beginning of a new word.

**Length**

**Purpose**

Returns the number of characters in field, including all spaces, numbers, and special characters.

**Format**

Length(field)

**Parameters**

field - any text, number, date, time, timestamp, or container field, or any text expression or numeric expression

**Data type returned**

number

**Description**

This function returns the number of characters in a specified field. 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

**Purpose**
Returns all letters in text as lowercase.

**Format**
Lower(text)

**Parameters**
text - any text expression or text field

**Data type returned**
text

**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

**Purpose**
Extracts the numberOfCharacters from text, starting at the character position specified by start.

**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

**Examples**
Middle(“(408)555-9054”;2;3) returns 408.
Middle(PhoneNumber;2;3) returns 408 when the PhoneNumber field contains (408) 555-9054.
Middle(“abcdefghij”;5;2) returns ef.
Middle (Name; Position (Name; ” ”; 1; 1) + 1; 3) returns Smi, when the text field Name contains John Smith.

MiddleValues

**Purpose**
Returns a text result containing the specified `numberOfValues` in text, starting with `startingValue`.

**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 Design functions for information about literal text parameters.

**Data type returned**
text

**Description**
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

**Purpose**
Returns a text result containing the `numberOfWords` from `text`, beginning at `startingWord`.

**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

**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.

**Note** The ampersand (&) and hyphen (-) characters identify the beginning of a new word.

NumToJText

**Purpose**
Converts Roman numbers in `number` to Japanese text.

**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**
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 —二三
- 3 - Traditional-old-style Kanji character number 壹弐参

Examples

\[
\text{NumToJText}(123456789;2;0) \quad \text{returns} \quad 1\text{億}2345\text{万}6789
\]

\[
\text{NumToJText}(123456789;3;2) \quad \text{returns} \quad 一億二千三百四十五万六千七百八十九
\]

PatternCount

Purpose
Returns the number of occurrences of \text{searchString} in \text{text}.

Format
\text{PatternCount}(\text{text};\text{searchString})

Parameters
- \text{text} - any text expression or text field
- \text{searchString} - any text expression or text field representing the set of characters you want to find

Data type returned
number

Examples

\[
\text{PatternCount}("\text{Mississippi}";"\text{is}") \quad \text{returns} \quad 2.
\]

\[
\text{PatternCount}("\text{Mississippi}";"\text{issi}") \quad \text{returns} \quad 1 \text{ (the function isn't inclusive).}
\]

\[
\text{PatternCount} (\text{Attending};"\text{Guest}") \quad \text{returns} \quad 1 \text{ if the Guest checkbox is one of the items selected in the Attending field.}
\]
Position

**Purpose**
Returns the starting position of the specified occurrence of `searchString` in `text`.

**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**
This function is not case-sensitive. If `searchString` isn’t contained in `text` or if there was no specified occurrence, zero is returned.

**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;" ";1;1)-1) returns Smith.
```

Proper

**Purpose**
Returns the first letter of each word in `text` as uppercase and all other letters as lowercase.

**Format**
```
Proper(text)
```
**Proper**

**Parameters**
- text - any text expression or text field

**Data type returned**
- text

**Examples**
- `Proper("ABCD")` returns **Abcd**.
- `Proper(Name)` returns **Yumiko Kitagawa**, when the Name field contains **YUMIKO KITAGAWA**.

**Quote**

**Purpose**
- Returns the text form of `text` enclosed in quotation marks.

**Format**
- `Quote(text)`

**Parameters**
- `text` - any text expression or field

**Data type returned**
- text

**Description**
- This function protects text from being evaluated by the `Evaluate` function. Special characters within `text` are escaped appropriately.

**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**

**Purpose**
- Replaces a string of characters in `text` with `replacementText`. 
**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**

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**

**Purpose**

Returns the specified numberOfCharacters in text, counting from the 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
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

Purpose
Returns a text result containing the specified number of values in text, counting from the right.

Format
RightValues(text;numberOfValues)

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

Important See Design functions for information about literal text parameters.

Data type returned
text

Description
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
RightWords

**Purpose**
Returns a text result containing the `numberOfWords` in `text`, counting from the right.

**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

**Examples**
RightWords("Plaid Canvas Suitcase";2) returns Canvas Suitcase.
RightWords(Name;1) returns Virtanen, when the Name field contains Matti Virtanen.

**Note** The ampersand (&) and hyphen (-) characters identify the beginning of a new word.

RomanHankaku

**Purpose**
Converts from Zenkaku alphanumerical and symbols to Hankaku alphanumerical and symbols.

**Format**
RomanHankaku(text)

**Parameters**
text - any text expression or text field

**Data type returned**
text

**Examples**
RomanHankaku("Ma c i n t o s h") returns Macintosh.
RomanZenkaku

**Purpose**
Converts from Hankaku alphanumeric and symbols to Zenkaku alphanumeric and symbols.

**Format**

RomanZenkaku(text)

**Parameters**

text - any text expression or text field

**Data type returned**
text

**Examples**

RomanZenkaku("Macintosh") returns M a c i n t o s h.

SerialIncrement

**Purpose**
Returns the combined text and numbers specified by text, with the numbers in text incremented by the specified amount.

**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**
This function doesn’t remove the text in text, which 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

**Purpose**
Returns a text string with every occurrence of `searchString` in `text` replaced by `replaceString` in `text`.

**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**
This 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

**Purpose**
Returns text stripped of all leading and trailing spaces.

**Format**
Trim(text)

**Parameters**
- text: any text expression or text field

**Data type returned**
text

**Description**
Use this 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

**Purpose**
Returns a copy of text with all leading and trailing spaces removed.

**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**
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, 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 `trimType`, as given in the following tables:

<table>
<thead>
<tr>
<th>This 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**

`TrimAll(名前,1,C)` returns 山田太郎 if the value of 名前 field is 山田 太郎

`TrimAll(ファイルメーカーPro は高品質*,1,C )` returns ファイルメーカーProは高品質
Upper

**Purpose**
Returns all letters in text as uppercase.

**Format**
Upper(text)

**Parameters**
text - any text expression or text field

**Data type returned**
text

**Description**
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

**Purpose**
Returns a count of the total number of values in text.

**Format**
ValueCount(text)

**Parameters**
text - any text expression or text field

**Important** See Design functions for information about literal text parameters.

**Data type returned**
number

**Description**
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**

**Purpose**

Returns a count of the total number of words in text.

**Format**

WordCount(text)

**Parameters**

text - any text expression or text field

**Data type returned**

number

**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.
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 quotations)
• 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</td>
<td>A number obtained by combining the red, green, and blue values to represent a color.</td>
</tr>
<tr>
<td>TextColor</td>
<td>The color of text to the color specified by the RGB function.</td>
</tr>
<tr>
<td>TextColorRemove</td>
<td>Text with the font colors reverted to the default font color for the field.</td>
</tr>
<tr>
<td>TextFont</td>
<td>Text in the specified font and character set.</td>
</tr>
<tr>
<td>TextFontRemove</td>
<td>Text with the fonts reverted to the default font for the field.</td>
</tr>
<tr>
<td>TextFormatRemove</td>
<td>Text with the formatting reverted to the default text format for the field.</td>
</tr>
<tr>
<td>TextSize</td>
<td>Text in the specified font size.</td>
</tr>
<tr>
<td>TextSizeRemove</td>
<td>Text with the font sizes reverted to the default font size for the field.</td>
</tr>
<tr>
<td>TextStyleAdd</td>
<td>Text with the specified styles added in a single action.</td>
</tr>
<tr>
<td>TextStyleRemove</td>
<td>Text with the specified styles removed in a single action.</td>
</tr>
</tbody>
</table>

RGB

**Purpose**

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.

**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**

color
**Description**
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 \( 256^2 = 65536 \)

**Tip** To determine the RGB value of a color:
- (Windows) In Layout mode, click the Fill Color palette in the formatting bar and choose Other Color. Values are shown for each of the basic colors.
- (Mac OS) Start the DigitalColor Meter application in the Applications/Utilities folder. Choose RGB As Actual Value, 8-bit. Move the pointer over colors onscreen to see their values.

**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**

**Purpose**
Changes the color of `text` to the color specified by the `RGB` function.

**Format**
`TextColor(text;RGB(red;green;blue))`

**Parameters**
- `text` - any text expression or text field
- `RGB(red;green;blue)` - any integer 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**
Use this function to change the color of `text`.

**Note** Text formatting options will be lost if the field type that is returned is something other than text.

**Tip** To determine the RGB value of a color:
- (Windows) In Layout mode, click the Fill Color palette in the formatting bar and choose **Other Color**. Values are shown for each of the basic colors.
- (Mac OS) Start the DigitalColor Meter application in the Applications/Utilities folder. Choose **RGB As Actual Value, 8-bit**. Move the pointer over colors onscreen to see their values.

**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**

**Purpose**
Removes all font colors in text, or removes the font color specified by the RGB function.

**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**
Use this function to revert text to the default font color for the field. If you don’t use the RGB function to 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 field 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

**Purpose**
Changes the font of text to the specified **fontName** or optional **{fontScript}**.

**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**
Spellings for font names must be correct and are case-sensitive. Text formatting options will be lost if the field 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**

**Purpose**
Removes all fonts in text, or removes the font specified by fontToRemove or the combination of fontToRemove and fontScript.

**Format**
TextFontRemove(text{;fontToRemove;fontScript})

**Parameters**
text - any text expression or text field.

fontToRemove - 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**
Use this function to revert text to the default for the field. 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 fontToRemove or the combination of fontToRemove 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 field 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
Text formatting functions

- 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

Purpose
Removes all text formatting from text in a single action.

Format
TextFormatRemove(text)

Parameters
text - any text expression or text field

Data type returned
text

Description
Use this function to remove all fonts, styles, font sizes, and font colors from the specified text.

Examples
TextFormatRemove("Plaid") returns the word Plaid without any text formatting applied.

TextSize

Purpose
Changes the font size of text to fontSize.
Text formatting functions

**Format**

TextSize(text;fontSize)

**Parameters**

text - any text expression or text or number field

fontSize - any font size expressed as an integer

**Data type returned**

text, number

**Description**

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 or number.

**Examples**

TextSize("Plaid";18) returns the word Plaid in 18 point text.

TextSize("Plaid";24) returns the word Plaid in 24 point text.

**TextSizeRemove**

**Purpose**

Removes all font sizes in text, or removes the font size specified by sizeToRemove.

**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**

Use this function to revert text to the default font size for the field. 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 field type that is returned is something other than text.
Text formatting functions

Examples

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

Purpose

Adds the specified styles to text in a single action.

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

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 field 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
Text formatting functions

- 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, to find every occurrence of several words and change their style, use the Substitute function combined with the TextStyleAdd function.

Substitute(ArticleBody;["Phrase1";TextStyleAdd("Phrase 1";Italic)];//"Phrase 2";TextStyleAdd("Phrase 2";Bold));)

TextStyleRemove

Purpose
Removes the specified styles from text in a single action.

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
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 field 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.
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</td>
<td>A number representing the number of hours in time.</td>
</tr>
<tr>
<td>Minute</td>
<td>A number representing the number of minutes in time.</td>
</tr>
<tr>
<td>Seconds</td>
<td>A number representing the number of seconds in time.</td>
</tr>
<tr>
<td>Time</td>
<td>A time result with the specified number of hours, minutes, and seconds.</td>
</tr>
</tbody>
</table>

**Hour**

**Purpose**

Returns a number representing the number of hours in time.

**Format**

Hour(time)

**Parameters**

time - any time value or field of type time

**Data type returned**

number

**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**

**Purpose**

Returns a number representing the number of minutes in time.

**Format**

Minute(time)
Parameters
time - any time value or field of type time

Data type returned
number

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

Purpose
Returns a number representing the number of seconds in time.

Format
Seconds(time)

Parameters
time - any time value or field of type time

Data type returned
number

Examples
Seconds("12:15:23") returns 23.
Hour(Duration) + (Minute(Duration)/60) + (Seconds(Duration)/3600 returns 2.504166, if the Duration time field contains 2:30:15.

Time

Purpose
Returns a time result with the specified number of hours, minutes, and seconds.

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
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.
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><strong>Timestamp</strong></td>
<td>A timestamp containing a calendar date and time of day.</td>
</tr>
</tbody>
</table>

**Timestamp**

**Purpose**

Returns a timestamp containing date as a calendar date and time as a time of day.

**Format**

Timestamp(date;time)

**Parameters**

date - any calendar date or date field
time - any time value or time field

**Data type returned**

timestamp

**Description**

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 your operating system.

**Examples**

Timestamp(Date(10;11;2008);Time(9;10;30)) returns 10/11/2008 9:10:30AM.
Timestamp(Date(10;11;2008);Time(13;10;30)) returns 10/11/2008 1:10:30PM.
Timestamp(Date(10;11;2008);Time(10;65;5)) returns 10/11/2008 11:05:05AM.
Timestamp(Date(10;35;2008);Time(4;5;6)) returns 11/4/2008 4:05:06AM.
Trigonometric functions

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>Acos</td>
<td>The arccosine, or inverse cosine, of a number.</td>
</tr>
<tr>
<td>Asin</td>
<td>The arcsine, or inverse sine, of a number.</td>
</tr>
<tr>
<td>Atan</td>
<td>The trigonometric arc tangent, or inverse tangent, of a number.</td>
</tr>
<tr>
<td>Cos</td>
<td>The cosine of the specified angle.</td>
</tr>
<tr>
<td>Degrees</td>
<td>Degrees, converted from the specified radians.</td>
</tr>
<tr>
<td>Pi</td>
<td>The value of the constant Pi, which is approximately 3.14159.</td>
</tr>
<tr>
<td>Radians</td>
<td>Radians, converted from the specified degrees.</td>
</tr>
<tr>
<td>Sin</td>
<td>The sine of the specified angle.</td>
</tr>
<tr>
<td>Tan</td>
<td>The tangent of the specified angle.</td>
</tr>
</tbody>
</table>

**Acos**

**Purpose**
Returns the arccosine (Acos), or inverse cosine, of a number.

**Format**
Acos (number)

**Parameters**
number - any numeric expression or field containing a numeric expression in the range -1 to 1

**Data type returned**
number

**Description**
The arccosine is the angle whose cosine is number. The returned angle is given in radians in the range 0 (zero) to Pi. The input number parameter must be between -1 and 1.

If you want to convert the result from radians to degrees, multiply it by 180/Pi or use the Degrees function.

**Examples**
Acos(-0.5) returns 2.0943951.
Acos(-0.5)*180/Pi returns 120.
Degrees(Acos(-0.5)) returns 120.
Acos(2.0) returns ? (not a number).

Asin

**Purpose**
Returns the arcsine (Asin), or inverse sine, of number.

**Format**
Asin (number)

**Parameters**
number - any numeric expression or field containing a numeric expression in the range -1 to 1

**Data type returned**
number

**Description**
The arcsine is the angle whose sine is number. The returned angle is given in radians in the range -Pi/2 to Pi/2. The input number parameter must be between -1 and 1.
To express the arcsine in degrees, multiply the result by 180/Pi or use the Degrees function.

**Examples**
Asin(-0.5) returns -0.523598776.
Asin(-0.5)*180/Pi returns -30.
Degrees(Asin(-0.5)) returns -30.
Asin(2) returns ? (not a number).

Atan

**Purpose**
Returns the trigonometric arc tangent (Atan), or inverse tangent, of number.

**Format**
Atan(number)

**Parameters**
number - any numeric expression or field containing a numeric expression
Data type returned
number

Description
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

Purpose
Returns the cosine (\(\text{Cos}\)) of angleInRadians.

Format
Cos(angleInRadians)

Parameters
angleInRadians - any numeric expression or field containing a numeric expression, in radians

Data type returned
number

Examples
Cos(1.047) returns \(0.50017107\)....
Cos(Radians(60)) returns \(0.5\).

Degrees

Purpose
Converts angleInRadians to degrees.

Format
Degrees(angleInRadians)

Parameters
angleInRadians - any numeric expression or field containing a numeric expression, in radians
**Data type returned**
number

**Description**
Use this function to translate results of 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**

**Purpose**
Calculates the value of the constant Pi (\(\pi\)), which is approximately 3.14159.

**Format**
Pi

**Parameters**
None

**Data type returned**
number

**Examples**
Pi * 15 returns 47.124.

**Radians**

**Purpose**
Converts angleInDegrees to radians.

**Format**
Radians(angleInDegrees)

**Parameters**
angleInDegrees - any numeric expression or field containing a numeric expression, in degrees
Data type returned
number

Description
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 \(0.78539816...\).
\(\sin(\text{Radians}(30))\) returns \(0.5\).

Sin

Purpose
Returns the sine (\(\sin\)) of \(\text{angleInRadians}\) expressed in radians.

Format
\(\sin(\text{angleInRadians})\)

Parameters
angleInRadians - any numeric expression or field containing a numeric expression, in radians

Data type returned
number

Examples
\(\sin(\text{Radians}(60))\) returns \(0.86602\).
\(\sin(0.610865)\) returns \(0.57357624...\).

Tan

Purpose
Returns the tangent (\(\tan\)) of \(\text{angleInRadians}\).

Format
\(\tan(\text{angleInRadians})\)
Parameters

angleInRadians - any numeric expression or field containing a numeric expression, in radians

Data type returned

number

Description

Use this function to calculate the Tan of angleInRadians.

Note  With the Tan function, you cannot use values exactly equal to 90 degrees (Pi/2 radians), or multiples of 90 degrees.

\[
\tan \theta = \frac{\sin \theta}{\cos \theta}
\]

Examples

\[
\tan(0.13) \text{ returns } 0.13073731...
\]

\[
\tan(\text{Radians}(34)) \text{ returns } 0.6745085.
\]