MMQuery

MMQuery_AddField

Description

This function Adds a new Field to an existing Table. The first parameter is the Name of a Table Occurrence of the Table to Add the new Field to. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the current Table and Add the Field to that Table. The second parameter is the name for the new Field. The third parameter is the Type of Field to Add and can be any one of the following values: "Text", "Number", "Date", "Time", "Timestamp", or "Container". If you do not specify the Field Type, MMQuery will create a "Text" Field. If you are using FileMaker Pro 8.5 or above, you can use the fourth parameter to specify if the new Field should be a Global Field. By default, MMQuery will check to make sure the Table Name you specify exists before attempting to Add the new Field. If you already know for sure that the Table exists, you can pass True as the fifth parameter.

Note: In FileMaker Pro 8.5 and above, you must insert a "Pause" Script Step after using this function in order for the new Field to be Added.

Return Type

Text

Format

MMQuery_AddField (**TOName** ; **FieldName** ; **FieldType** ; **AsGlobal** ; **SkipTONameCheck**)

Required Parameters

TOName
The Name of a Table Occurrence for the Table to receive the new Field. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database FileName (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldName
The Name of the new Field.

Optional Parameters

FieldType
The Type of Field to Add. Specify "Text", "Number", "Date", "Time", "Timestamp", or "Container". MMQuery uses "Text" as the default.

AsGlobal
If True, the plug-in will Add a Field with Global storage instead of a normal Field. This parameter is only available in FileMaker Pro 8.5 and above.

SkipTONameCheck
If True, the plug-in will not check to make sure the TO Name you specified actually exists before attempting to Add the Field.

Related Items

MMQuery_FieldExists, MMQuery_GetFieldIDs, MMQuery_GetFieldNames, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByFind, MMQuery_GetFieldValueByID, MMQuery_RemoveField, MMQuery_SetFieldValueByFind, MMQuery_SetFieldValueByID

Example 1

Code:

```
MMQuery_AddField( "MyTable" ; "MyField" )
```

Result:

Adds a new Text Field named "MyField" to the "MyTable" Table.

Example 2

Code:

```
MMQuery_AddField( "" ; "gNum" ; "Number" ; True )
```

Result:

Adds a new Global Number Field named "gNum" to the current Table. (This would only work on FileMaker Pro 8.5 and above.)

Example 3

Code:

```
MMQuery_AddField( "AnotherFile.MyTable" ; "MyField" )
```

Result:

In FileMaker Pro 11 and above, use this form to add a field to a Table Occurrence in a separate, open Database file.

MMQuery_AddRecord

Description

This function Adds a new Record to an existing Table. The first parameter is the Name of a Table Occurrence of the Table to receive the new Record. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the current Table and Add the Record to that Table. The second parameter is a return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the Field Values for the new Record. If a Field Value needs a return in it, you can use "<CR>" to indicate the return. By default, MMQuery will check to make sure the Table Name and all the Field Names you specify exist before attempting to Add the new Record. If you already know for sure that the Table Name and the Field Names exist, you can pass True as the third and fourth parameters.

Return Type

Text

Format

MMQuery_AddRecord (**TOName** ; **FieldValues** ; **SkipTONameCheck** ; **SkipFieldNameChecks**)

Required Parameters

TOName
The Name of a Table Occurrence for the Table to receive the new Record. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldValues
A return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the Field Values for the new Record. Use "" to indicate a return or new line in a Field Value.

Optional Parameters

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before attempting to Add the Record.

SkipFieldNameChecks
If True, MMQuery will not check to make sure all the Field Names in the FieldValues parameter actually exist before attempting to Add the Record.

Related Items

MMQuery_FindRecords, MMQuery_RemoveRecordByFind, MMQuery_RemoveRecordByID, MMQuery_UpdateRecordByFind, MMQuery_UpdateRecordByID

Examples

Example 1

Code:

```
MMQuery_AddRecord( "MyTable" ; "MyField=Some value¶AnotherField=Some other value" )
```

Result:

Adds a new Record to the "MyTable" Table setting the "MyField" and "AnotherField" Fields.

Example 2

Code:

```
MMQuery_AddRecord( "" ; "MyField=A<CR>value<CR>with<CR>multiple<CR>lines" )
```

Result:

Adds a new Record to the current Table setting the "MyField" field with a multi-line value.

Example 3

Code:

```
MMQuery_AddRecord( "AnotherFile.MyTable" ; "MyField=Some value" )
```

Result:

In FileMaker Pro 11 and above, use this form to add a new Record to a Table Occurrence in a separate, open Database file.

MMQuery_AddTable

Description

This function Adds a new Table to the current Database File. The first parameter is the Name of the new Table, which must be unique for the current Database File. The second parameter is a return- or paragraph mark-separate list of "FieldName=FieldType" pairs used to describe the Fields that should be Added to the new Table. Valid Field Types are "Text", "Number", "Date", "Time", "Timestamp", and "Container". If you are using FileMaker Pro 8.5 or above, you can also specify the following Global Field Types: "GlobalText", "GlobalNumber", "GlobalDate", "GlobalTime", "GlobalTimestamp", and "GlobalContainer".

Note: In FileMaker Pro 8.5 and above, you must insert a "Pause" Script Step after using this function in order for the new Table to be Added.

Return Type

Text

Format

MMQuery_AddTable (**TableName** ; **FieldDefinitions**)

Required Parameters

TableName
The Name of the Table to Add. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TableName".

FieldDefinitions
A return- or paragraph mark-separated list of "FieldName=FieldType" pairs describing the Field Names for the new Table. Use "Text", "Number", "Date", "Time", "Timestamp", "Container", "GlobalText", "GlobalNumber", "GlobalDate", "GlobalTime", "GlobalTimestamp", and "GlobalContainer" as the Field Types.

Related Items

MMQuery_GetBaseTableNames, MMQuery_GetTOIDs, MMQuery_GetTONames, MMQuery_RemoveTable, MMQuery_TOExists

Examples

Example 1

Code:

```
MMQuery_AddTable( "MyNewTable" ; "AField=Text¶BField=Number" )
```

Result:

Adds a new Table to the current Database File containing two fields named "AField" and "BField" which have the Field Types Text and Number respectively.

Example 2

Code:

```
MMQuery_AddTable( "AnotherFile.MyNewTable" ; "AField=Text" )
```

Result:

In FileMaker Pro 11 and above, use this form to add a new Table in a separate, open Database file.

MMQuery_Configure

Description

Calling this function with no parameters will open the MMQuery Configuration Dialog. You can optionally open to a specific tab by specifying the name of the tab as the "Option" parameter. This function also allows you to get or set any preference found in the Configuration Dialog.

Valid PrefNames:

"AddHelp" - If True, Function specific Help will be added to Calculations when inserting the Plug-in's Functions.
"AlwaysUseFileSQL" - If True, and if running under FileMaker Pro 11+, the plug-in will always use the newer, more strict/more complete, SQL Interface.

Return Type

Varies

Format

MMQuery_Configure (**Option** ; **PrefName** ; **PrefValue**)

Optional Parameters

Option
Specify the name of a Tab in the Configuration Dialog to show it opened to that tab.
Specify "Get" with the PrefName parameter to get a preference value. (If not found, and PrefValue is defined, PrefValue will be returned.)
Specify "Set" with the PrefName and PrefValue parameters to set a preference value.

PrefName
The Name of the Preference to Get or Set. (See the Function Description for a list of valid PrefNames.)

PrefValue
The Value of the Preference to Set. (See the Function Description for some possible values.)

Examples

Example 1

Code:

```
MMQuery_Configure
```

Result:

Opens the MMQuery Configuration Dialog. (Because the "Option" parameter is not used, the dialog will open to the "Basics" tab.)

Example 2

Code:

```
MMQuery_Configure( "About" )
```

Result:

Opens the MMQuery Configuration Dialog to the "About" tab.

Example 3

Code:

```
MMQuery_Configure( "Get" ; "AddHelp" )
```

Result:

Returns the value of the 'Add Help Comments to External Functions' setting from the Configuration Dialog.

Example 4

Code:

```
MMQuery_Configure( "Set" ; "AlwaysUseFileSQL" ; "True" )
```

Result:

Sets the 'Always use new SQL Interface' setting in the Configuration Dialog to true.

MMQuery_EscapeQuotes

Description

This is a utility function that takes raw Text and conforms it to what the ExecuteSQL function needs in order to function correctly. It turns any Backward Slashes (\ ) into a double-Backward Slash (\\ ) and turns any Single Quotes (') into Backward Slash-Single Quote (\').

Return Type

Text

Format

MMQuery_EscapeQuotes (**Text**)

Required Parameters

Text
The Text to conform to what ExecuteSQL needs.

Related Items

MMQuery_ExecuteSQL, MMQuery_ExecuteSQLEx

Example

Code:

```
MMQuery_EscapeQuotes( "I can't seem to find the backslash (\) on my keyboard." )
```

Result:

Returns "I can\'t seem to find the backslash (\\) on my keyboard."

MMQuery_ExecuteSQL

Description

This function allows you to use SQL statements to run queries against the current Database File. If the Query you are using will be returning Field values and multiple Records, you can optionally define the Separators that should be placed between the Fields and Records using the last two parameters. By default, MMQuery will use a comma between multiple Fields and a return between multiple Records. The Field and Record Separators you specify are not limited to a single character.

Return Type

Text

Format

MMQuery_ExecuteSQL (**SQL** ; **FieldSeparator** ; **RecordSeparator**)

Required Parameters

SQL
The SQL statement to evaluate against the current Database File.

Optional Parameters

FieldSeparator
The Separator to use between multiple Fields when doing an SQL Select. By default, Fields will be separated with a comma.

RecordSeparator
The Separator to use between multiple Records when doing an SQL Select. By default, Records will be separated with a return.

Related Items

MMQuery_EscapeQuotes, MMQuery_ExecuteSQLEx, MMQuery_GetAsSQLDate, MMQuery_GetAsSQLTime, MMQuery_GetAsSQLTimestamp

Examples

Example 1

Code:

```
MMQuery_ExecuteSQL( "select * from MyTable" )
```

Result:

Returns every field and every record from the "MyTable" Table.

Example 2

Code:

```
MMQuery_ExecuteSQL( "select '<tr><td>' + \"First\", \"Last\" + '</td></tr>' from MyTable"
```

Result:

Returns something like "<tr><td>Mary</td><td>Jones</td></tr>" for every record in the "MyTable" Table.

MMQuery_ExecuteSQLEx

Description

Note: This function is only available in FileMaker Pro 11 and above.

This "extended" version of MMQuery_ExecuteSQL allows you to optionally specify a separate, open Database File from the current Database File. The third parameter allows you to specify that the plug-in store the results internally, which you can then retrieve the value from individual cells using the MMQuery_GetRecordData function. Retrieving the data in this manner allows you to retrieve not just textual values, but any of the FileMaker Pro Data Types including Container data. On the other hand, if you want the results returned as text, you can optionally define the Separators that should be placed between the Fields and Records using the last two parameters. By default, MMQuery will use a comma between multiple Fields and a return between multiple Records. The Field and Record Separators you specify are not limited to a single character. Finally, this function allows you to use "SQL Parameters" (question marks in your query) and then specify those parameters starting with the sixth parameter position. This allows you to create queries that can copy Stored Container data into other fields, among other things.

Return Type

Text

Format

MMQuery_ExecuteSQLEx (**FileName** ; **SQL** ; **UseRecordData** ; **FieldSeparator** ; **RecordSeparator** ; **...**)

Required Parameters

FileName
The FileName (without the ".fp7" extension) of the open Database File you are wanting to run this SQL query against. Specify "" to use the Current File.

SQL
The SQL statement to evaluate.

Optional Parameters

UseRecordData
If True, this function will return a blank response when successful and you can use the MMQuery_GetRecordData function to retrieve any data. By default, this is False.

FieldSeparator
The Separator to use between multiple Fields when doing an SQL Select. By default, Fields will be separated with a comma.

RecordSeparator
The Separator to use between multiple Records when doing an SQL Select. By default, Records will be separated with a return.

...
Any number of extra parameters corresponding to any ?'s in your SQL query.

Related Items

MMQuery_EscapeQuotes, MMQuery_ExecuteSQL, MMQuery_GetAsSQLDate, MMQuery_GetAsSQLTime, MMQuery_GetAsSQLTimestamp, MMQuery_GetRecordData, MMQuery_GetRecordDataColumnCount, MMQuery_GetRecordDataRowCount

Examples

Example 1

Code:

```
MMQuery_ExecuteSQLEx( "" ; "select * from MyTable" ; True )
```

Result:

Returns an empty response (meaning success), and populates the internal Record Data with every field and every record from the "MyTable" Table Occurrence in the current Database File.

Example 2

Code:

```
MMQuery_ExecuteSQLEx( "Contacts" ; "select '<tr><td>' + \"First\", \"Last\" + '</td></tr>'
```

Result:

Returns something like "<tr><td>Mary</td><td>Jones</td></tr>" for every record in the "People" Table Occurrence in the "Contacts" Database File.

Example 3

Code:

```
MMQuery_ExecuteSQLEx( "" ; "update People set \"Photo\"=?" ; False ; "," ; "¶" ; SomeConta
```

Result:

Sets the "Photo" field in the "People" TO with the value from the "SomeContainerField" field.

MMQuery_FieldExists

Description

This function checks for the Existence of a Field in an existing Table. The first parameter is the Name of a Table Occurrence of the Table to look at for the Field. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the current Table to look at for the Field. The second parameter is the Name of the Field to look for. By default, MMQuery will check to make sure the TO Name you specify exists before searching it for the Field. If you already know for sure that the TO exists, you can pass True as the third parameter.

Return Type

Number (1=True, 0=False)

Format

MMQuery_FieldExists (**TOName** ; **FieldName** ; **SkipTONameCheck**)

Required Parameters

TOName
The Name of the Table Occurrence to search. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldName
The Name of the Field to search for.

Optional Parameters

SkipTONameCheck
If True, the plug-in will not check to make sure the TO Name you specified actually exists before searching it for the Field.

Related Items

MMQuery_AddField, MMQuery_GetFieldIDs, MMQuery_GetFieldNames, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByFind, MMQuery_GetFieldValueByID, MMQuery_RemoveField, MMQuery_SetFieldByFind, MMQuery_SetFieldByID

Examples

Example 1

Code:

```
MMQuery_FieldExists( "MyTable" ; "AField" )
```

Result:

Returns 1 when a Field Named "AField" exists in a Table Named "MyTable".

Example 2

Code:

```
MMQuery_FieldExists( "" ; "DoesntExist" )
```

Result:

Returns 0 when a Field Named "DoesntExist" does not exist in the current Table.

Example 3

Code:

```
MMQuery_FieldExists( "AnotherFile.MyTable" ; "AField" )
```

Result:

In FileMaker Pro 11 and above, use this form to add a check for a field in a Table Occurrence in a separate, open Database file.

MMQuery_FindRecords

Description

This function does a Find against a Table and returns all matches. The first parameter is the Name of the Table Occurrence to perform the Find in. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the current TO to perform the Find in. The second parameter is a return- or paragraph mark-separated list of "FieldName=FieldValue" pairs that describes the Criteria for Finding the Records you want. See the "Finding What You Want" section below for help in defining your Find Criteria. The FieldNamesToReturn parameter is a return- or paragraph mark-separated list of Fields you want returned in the Result. In other words, if you have 10 fields in your Table, but you are only interested in the values of the "FirstName" and "LastName" fields, you would specify "FirstName¶LastName" as the FieldNamesToReturn parameter. (Note: You can specify fields in the FindCriteria other than what you want returned in the result.) The fourth parameter is a return- or paragraph mark-separated list of "FieldNames=SortDirection" pairs that define how to Sort the Result. For the Sort Direction you can use "Asc" or "Desc" to sort Ascending or Descending respectively. By default, MMQuery will use a comma between multiple Fields and a return between multiple Records. If you need to use different Separators, you can use the fifth and sixth parameters. The Field and Record Separators are not limited to a single character. By default, MMQuery will check to make sure the TO Name and all the Field Names you specify exist before attempting to perform the Find. If you already know for sure that the TO Name and the Field Names exist, you can pass True as the seventh and eighth parameters. If you want the Find to be Case Sensitive, you can specify True as the ninth parameter. Finally, if you are running FileMaker Pro 11 or above, you can use the tenth parameter to specify that the function store the results internally to be retrieved later using the MMQuery_GetRecordData function.

Finding What You Want

The FindCriteria parameter in this function and other related functions allows you to define exactly which Records in your Table that you want to work with. Not every Find you want to do is as simple as saying, "This field needs this exact value in it". So, this section explains how to do more complex Finds to get exactly what you want. First, let's say we have a table with the following values in it:

Record	FirstName	LastName	Colors
1	Mary	Jones	Red Blue
2	LeAnn	Johnson	Green-Yellow Red
3	Ann	Parsons	Green Orange

Now, the following are different FindCriteria parameters you can use and what MMQuery would return for each:

"FirstName=Mary" - This would return Record number 1. Specifying a single Value for a Field will return any Record where that Field's Value is exactly what you specified.

"FirstName=Mary¶Ann" - This would return Record numbers 1 and 3. Specifying multiple Values for a Field will return any Record where that Field's Value is exactly one of those Values. Note that this does not match the value "LeAnn" in the second record.

"::Contains::FirstName=Ann" - This would return Record numbers 2 and 3. Using the special "::Contains::" keyword before the Field Name tells MMQuery to match the Value anywhere in the Field instead of matching the Field Value exactly.

"::Begins-With::LastName=J" - This would return Record numbers 1 and 2. Using the special "::Begins-With::" keyword before the Field Name tells MMQuery to find any Record whose Field Value starts with the Value you specify.

"::Ends-With::FirstName=n" - This would return Record numbers 2 and 3. Using the special "::Ends-With::" keyword before the Field Name tells MMQuery to find any Record whose Field Value ends with the Value you specify.

"Colors=Blue" - This would not return any Records because none of the Records have the exact value "Blue" for the "Colors" field.

"::Contains::Colors=Blue¶Green" - This would return Record numbers 1, 2 and 3. Since the "::Contains::" keyword will tell MMQuery to find the Value anywhere in the Field, this will match "Green-Yellow" in the second Record.

"::Contains-Value::Colors=Blue¶Green" - This would return Record numbers 1 and 3. The special "::Contains-Value::" keyword before the Field Name tells MMQuery to find exact Values in the Field. (A Value is defined here as any Text that is on a line by itself.)

Return Type

Text

Format

MMQuery_FindRecords (**TOName** ; **FindCriteria** ; **FieldNamesToReturn** ; **SortFields** ; **FieldSeparator** ; **RecordSeparator** ; **SkipTONameCheck** ; **SkipFieldNameChecks** ; **CaseSensitive** ; **UseRecordData**)

Required Parameters

TOName
The Name of a Table Occurrence of the Table to perform the Find in. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FindCriteria
A return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the Criteria for the Find. See the "Finding What You Want" section above for help in setting up the Find Criteria.

Optional Parameters

FieldNamesToReturn
A return- or paragraph mark-separated list of Field Names that you want returned from the Find. If you do not specify this parameter, all Fields will be returned.

SortFields
A return- or paragraph mark-separated list of "FieldName=SortDirection" pairs describing the Sort order for the returned data. Use "Asc" or "Desc" for the Sort Direction to indicate Ascending or Descending respectively.

FieldSeparator
The Separator to use between multiple Fields. By default, Fields will be separated with a comma.

RecordSeparator
The Separator to use between multiple Records. By default, Records will be separated with a return.

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before performing the Find.

SkipFieldNameChecks
If True, MMQuery will not check to make sure all the Field Names in the various parameters actually

exist before performing the Find.

CaseSensitive
If True, MMQuery will preserve the Case in the Field Values of the Find Criteria. (In other words, if CaseSensitive is True, "Green" would not match "green".)

UseRecordData
If True, this function will return a blank response when successful and you can use the MMQuery_GetRecordData function to retrieve any data. The default is False. (Note: This parameter is only available under FileMaker Pro 11 and above.)

Related Items

MMQuery_AddRecord, MMQuery_GetFieldValueByFind, MMQuery_GetRecordData, MMQuery_GetRecordDataColumnCount, MMQuery_GetRecordDataRowCount, MMQuery_RemoveRecordByFind, MMQuery_RemoveRecordByID, MMQuery_SetFieldByFind, MMQuery_SetFieldByID, MMQuery_UpdateRecordByFind, MMQuery_UpdateRecordByID

Examples

Example 1

Code:

```
MMQuery_FindRecords( "MyTable" ; "AField=Some Value" )
```

Result:

Returns all Records from the "MyTable" Table where the Field "AField" has the value "Some Value".

Example 2

Code:

```
MMQuery_FindRecords( "" ; "LastName=Jones<CR>Johnson" ; "LastName¶FirstName" ; "LastName=D
```

Result:

Returns all Records from the current Table where the "LastName" Field is set to either Jones or Johnson. The Result will only contain the LastName and FirstName fields (in that order) separated by a comma and a space, and Sorted Descending by Last Name and then First Name.

Example 3

Code:

```
MMQuery_FindRecords( "" ; "::Contains::Colors=Green" )
```

Result:

Returns all Records from the current Table where the word "Green" appears somewhere in the "Colors" Field.

Example 4

Code:

```
MMQuery_FindRecords( "" ; "::Begins-With::LastName=J" )
```

Result:

Returns all Records from the current Table where the "LastName" Field starts with the letter "J".

Example 5

Code:

```
MMQuery_FindRecords( "AnotherFile.MyTable" ; "AField=Some Value" ; "" ; "" ; "" ; "" ; ""
```

Result:

In FileMaker Pro 11 and above, use this form to Find Records in a Table Occurrence in a separate, open Database file and store the results internally to later be retrieved with MMQuery_GetRecordData.

MMQuery_GetAsSQLDate

Description

This function will convert a FileMaker Pro Date value into an SQL Date Format suitable for using in an SQL query.

Return Type

Text

Format

MMQuery_GetAsSQLDate (**FMPDate**)

Required Parameters

FMPDate
The FileMaker Pro Date to convert to an SQL formatted Date.

Related Items

MMQuery_GetAsSQLTime, MMQuery_GetAsSQLTimestamp

Example

Code:

```
MMQuery_GetAsSQLDate( MyDate )
```

Result:

Returns something like: 2010-03-29

MMQuery_GetAsSQLTime

Description

This function will convert a FileMaker Pro Time value into an SQL Time Format suitable for using in an SQL query.

Return Type

Text

Format

MMQuery_GetAsSQLTime (**FMPTime**)

Required Parameters

FMPTime
The FileMaker Pro Time to convert to an SQL formatted Time.

Related Items

MMQuery_GetAsSQLDate, MMQuery_GetAsSQLTimestamp

Example

Code:

```
MMQuery_GetAsSQLTime( MyTime )
```

Result:

Returns something like: 14:47:43

MMQuery_GetAsSQLTimestamp

Description

This function will convert a FileMaker Pro Timestamp value into an SQL Timestamp Format suitable for using in an SQL query.

Return Type

Text

Format

MMQuery_GetAsSQLTimestamp (**FMPTimestamp**)

Required Parameters

FMPTimestamp
The FileMaker Pro Timestamp to convert to an SQL formatted Timestamp.

Related Items

MMQuery_GetAsSQLDate, MMQuery_GetAsSQLTime

Example

Code:

```
MMQuery_GetAsSQLTimestamp( MyTimestamp )
```

Result:

Returns something like: 2010-03-29 14:48:56

MMQuery_GetBaseTableNames

Description

Note: This function is only available in FileMaker Pro 11 and above.

This function returns either the Base Table Name of a specific Table Occurrence or a list of "TOName=BaseTableName" pairs for all Table Occurrences in a Database. If you specify a TOName as the first parameter, it will return the Base Table Name of that Table Occurrence, otherwise it will return all Base Table Names. By default, MMQuery will check to make sure the Table Occurrence Name you specify exists before getting the Base Table Names. If you already know for sure that the TO Name exists, you can pass True as the second parameter.

Return Type

Text

Format

MMQuery_GetBaseTableNames (**TOName** ; **SkipTONameCheck**)

Optional Parameters

TOName
If you only want the Base Table Name of one Table Occurrence, specify its Name here. Specify "" to return the Base Table Names of all Table Occurrences. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) followed by a period to retrieve all the Base Table Names in that file (eg. "FileName."). Also in FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specify actually exists.

Related Items

MMQuery_GetTOIDs, MMQuery_GetTONames, MMQuery_TOExists

Examples

Example 1

Code:

```
MMQuery_GetBaseTableNames
```

Result:

Returns something like:<br/>
Students=People<br/>
Teachers=People

Example 2

Code:

```
MMQuery_GetFieldIDs( "AnotherFile." )
```

Result:

In FileMaker Pro 11 and above, use this form to get the Base Table Names of all Table Occurrences in a separate, open Database file.

Example 3

Code:

```
MMQuery_GetFieldIDs( "AnotherFile.MyTable" )
```

Result:

In FileMaker Pro 11 and above, use this form to get the Base Table Name of a Table Occurrence in a separate, open Database file.

MMQuery_GetFieldIDs

Description

This function returns either the ID of a specific Field or a list of "FieldName=FieldID" pairs for all Fields in a Table. The first parameter is the Name of a Table Occurrence of the Table to return the Field IDs from. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the current Table to return the Field IDs from. If you specify a Field Name in the second parameter, MMQuery will return the Field ID of that specific Field. If you do not specify this parameter, MMQuery will return a list of "FieldName=FieldID" pairs for all the Fields in the Table. By default, MMQuery will check to make sure the TO Name and the Field Name you specify exist before getting the Field IDs. If you already know for sure that the TO Name and the Field Name exist, you can pass True as the third and fourth parameters.

Note: In FileMaker Pro 7 through 10, to ensure that you get the Field IDs of all the Fields in your Table, you must make sure that there is not a Layout in your Database with the same name as the TO Name you specify for this function. In FileMaker Pro 11 and above, this is not an issue.

Return Type

Text or Number

Format

MMQuery_GetFieldIDs (**TOName** ; **FieldName** ; **SkipTONameCheck** ; **SkipFieldNameCheck**)

Optional Parameters

TOName
The Name of a Table Occurrence of the Table that contains the Fields. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldName
If you specify this parameter, MMQuery will return the ID of the Field you specify. Otherwise, MMQuery will return all the IDs of all the Fields.

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specify actually exists.

SkipFieldNameCheck
If True, MMQuery will not check to make sure the Field Name you specify actually exists.

Related Items

MMQuery_AddField, MMQuery_FieldExists, MMQuery_GetFieldNames, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByFind, MMQuery_GetFieldValueByID, MMQuery_RemoveField, MMQuery_SetFieldValueByFind, MMQuery_SetFieldValueByID

Examples

Example 1

Code:

```
MMQuery_GetFieldIDs( "MyTable" )
```

Result:

Returns something like:<br/>
AField=1<br/>
BField=2

Example 2

Code:

```
MMQuery_GetFieldIDs( "" ; "BField" )
```

Result:

Returns 2 if the Field ID for the "BField" Field in the current Table is 2.

Example 3

Code:

```
MMQuery_GetFieldIDs( "AnotherFile.MyTable" )
```

Result:

In FileMaker Pro 11 and above, use this form to get the Field IDs of a Table in a separate, open Database file.

MMQuery_GetFieldNames

Description

This function returns a list of Field Names from a Table. The first parameter is the name of a Table Occurrence of the Table to return the Field Names from. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the current Table to return the Field Names from. By default, MMQuery will check to make sure the TO Name you specify exists before getting the Field Names. If you already know for sure that the TO Name exists, you can pass True as the second parameter.

Note: In FileMaker Pro 7.0 to FileMaker Pro 8.0, to ensure that you get the Names of all the Fields in your Table, you must make sure that there is not a Layout in your Database with the same name as the TO Name you specify for this Function. If you are using FileMaker Pro 8.5 or above, this is no longer an issue.

Return Type

Text

Format

MMQuery_GetFieldNames (**TOName** ; **SkipTONameCheck**)

Optional Parameters

TOName
The Name of a Table Occurrence of the Table that contains the Fields. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specify actually exists.

Related Items

MMQuery_AddField, MMQuery_FieldExists, MMQuery_GetFieldIDs, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByFind, MMQuery_GetFieldValueByID, MMQuery_RemoveField, MMQuery_SetFieldValueByFind, MMQuery_SetFieldValueByID

Examples

Example 1

Code:

```
MMQuery_GetFieldNames( "MyTable" )
```

Result:

Returns something like:
AField
BField

Example 2

Code:

```
MMQuery_GetFieldNames( "AnotherFile.MyTable" )
```

Result:

In FileMaker Pro 11 and above, use this form to get the Field Names of a Table in a separate, open Database file.

MMQuery_GetFieldTypes

Description

This function returns either the Field Type of a specific Field or a List of "FieldName=FieldType" pairs for all Fields in a Table. The first parameter is the Name of a Table Occurrence of the Table to return the Field Types from. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout TO to return the Field Types from. If you specify a Field Name in the second parameter, MMQuery will return the Field Type of that specific Field. If you do not specify this parameter, MMQuery will return a list of "FieldName=FieldType" pairs for all the Fields in the Table. By default, MMQuery will check to make sure the TO Name and the Field Name you specify exist before getting the Field Types. If you already know for sure that the TO Name and Field name exist, you can pass True as the third and fourth parameters.

Note: In FileMaker Pro 7 through 10, to ensure that you get the Field Types of all the Fields in your Table, you must make sure that there is not a Layout in your Database with the same name as the TO Name you specify for this function. In FileMaker Pro 11 and above, this is no longer an issue.

Return Type

Text

Format

MMQuery_GetFieldTypes (**TOName** ; **FieldName** ; **SkipTONameCheck** ; **SkipFieldNameCheck**)

Optional Parameters

TOName
The Name of a Table Occurrence of the Table that contains the Fields. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldName
If you specify this parameter, MMQuery will return the Field Type of the Field you specify. Otherwise, MMQuery will return all the Field Type of all the Fields.

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specify actually exists.

SkipFieldNameCheck
If True, MMQuery will not check to make sure the Field Name you specify actually exists.

Related Items

MMQuery_AddField, MMQuery_FieldExists, MMQuery_GetFieldNames, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByFind, MMQuery_GetFieldValueByID, MMQuery_RemoveField, MMQuery_SetFieldValueByFind, MMQuery_SetFieldValueByID

Examples

Example 1

Code:

```
MMQuery_GetFieldTypes( "MyTable" )
```

Result:

Returns something like:<br/>
AField=Text<br/>
BField=Calculation (Number)<br/>
CField=Global Timestamp

Example 2

Code:

```
MMQuery_GetFieldTypes( "" ; "AField" )
```

Result:

Returns "Text" if the Field Type for the "AField" Field in the current Table is a Text Field.

Example 3

Code:

```
MMQuery_GetFieldTypes( "AnotherFile.MyTable" )
```

Result:

In FileMaker Pro 11 and above, use this form to get the Field Types of a Table in a separate, open Database file.

MMQuery_GetFieldValueByFind

Description

This function retrieves the Value of a single Field using Find Criteria. The first parameter is the Name of a Table Occurrence of the Table to Get the Field Value from. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout TO to Get the Field Value from. The second parameter is the Name of the Field to retrieve. The third parameter is a return- or paragraph mark-separated list of "FieldName=FieldValue" pairs that describes the Criteria for Finding the Record to Get the Field Value from. See the "Finding What You Want" section in the MMQuery_FindRecords function for help in defining your Find Criteria. By default, MMQuery will check to make sure the TO Name and all the Field Names you specify exist before attempting to Get the Field Value. If you already know for sure that the TO Name and the Field Names exist, you can pass True as the fourth and fifth parameters. Finally, if you want the Find to be Case Sensitive, you can specify True as the last parameter.

Return Type

Varies

Format

MMQuery_GetFieldValueByFind (**TOName** ; **FieldName** ; **FindCriteria** ; **SkipTONameCheck** ; **SkipFieldNameChecks** ; **CaseSensitive**)

Required Parameters

TOName
The Name of a Table Occurrence of the Table to Get the Field Value from. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldName
The Name of the Field to retrieve the Value from.

FindCriteria
A return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the Criteria for Finding the Record to get the Field Value from. See the "Finding What You Want" section of the MMQuery_FindRecords function for help in setting up the Find Criteria.

Optional Parameters

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before retrieving the Field Value.

SkipFieldNameChecks
If True, MMQuery will not check to make sure all the Field Names in the various parameters actually exist before retrieving the Field Value.

CaseSensitive
If True, MMQuery will preserve the Case in the Field Values of the Find Criteria. (In other words, if CaseSensitive is True, "Green" would not match "green".)

Related Items

MMQuery_AddField, MMQuery_FieldExists, MMQuery_FindRecords, MMQuery_GetFieldIDs, MMQuery_GetFieldNames, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByID, MMQuery_RemoveField, MMQuery_SetFieldValueByFind, MMQuery_SetFieldValueByID

Examples

Example 1

Code:

```
MMQuery_GetFieldValueByFind( "MyTable" ; "BField" ; "AField=some value" )
```

Result:

This would retrieve the Value of the "BField" Field from the "MyTable" Table from the Record where the "AField" Field has the Value "some value".

Example 2

Code:

```
MMQuery_GetFieldValueByFind( "" ; "LastName" ; "FirstName=Ann" )
```

Result:

This would retrieve the Value of the "LastName" Field from the current Table from the Record where the "FirstName" Field has the value "Ann".

Example 3

Code:

```
MMQuery_GetFieldValueByFind( "AnotherFile.MyTable" ; "BField" ; "AField=some value" )
```

Result:

In FileMaker Pro 11 and above, use this form to get the Field Value from a Table Occurrence in a separate, open Database file.

MMQuery_GetFieldValueByID

Description

This function retrieves the Value of a Single Field from a Record with a specific ID. The first parameter is the Name of a Table Occurrence of the Table to Get the Field Value from. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout TO to Get the Field Value from. The second parameter is the Name of the Field to retrieve. The third parameter is the Internal FileMaker Record ID of the Record to retrieve the Field Value from. If you do not specify this parameter, MMQuery will use Get(RecordID) to get the Record ID. By default, MMQuery will check to make sure the TO Name, the Field Name, and the Record ID you specify exist before attempting to Get the Field Value. If you already know for sure that the TO Name, the Field Name, and the Record ID exist, you can pass True as the fourth, fifth, and sixth parameters.

Return Type

Varies

Format

MMQuery_GetFieldValueByID (**TOName** ; **FieldName** ; **FMRecID** ; **SkipTONameCheck** ; **SkipFieldNameCheck** ; **SkipFMRecIDCheck**)

Required Parameters

TOName
The Name of a Table Occurrence for the Table to Get the Field Value from. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldName
The Name of the Field to retrieve the Value from.

Optional Parameters

FMRecID
The Internal FileMaker Record ID of the Record to Get the Field Value from.

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before retrieving the Field Value.

SkipFieldNameCheck
If True, MMQuery will not check to make sure the Field Name you specified actually exists before retrieving the Field Value.

SkipFMRecIDCheck
If True, MMQuery will not check to make sure the Record ID you specified actually exists before retrieving the Field Value.

Related Items

MMQuery_AddField, MMQuery_FieldExists, MMQuery_GetFieldIDs, MMQuery_GetFieldNames, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByFind, MMQuery_RemoveField, MMQuery_SetFieldValueByFind, MMQuery_SetFieldValueByID

Example 1

Code:

```
MMQuery_GetFieldValueByID( "MyTable" ; "AField" ; 4 )
```

Result:

This would retrieve the Value of the "AField" Field from the "MyTable" Table from the Record with the ID 4.

Example 2

Code:

```
MMQuery_GetFieldValueByFind( "" ; "FirstName" )
```

Result:

This would retrieve the Value of the "FirstName" Field from the current Table from the current Record.

Example 3

Code:

```
MMQuery_GetFieldValueByID( "AnotherFile.MyTable" ; "AField" )
```

Result:

In FileMaker Pro 11 and above, use this form to get the Field Value from a Table Occurrence in a separate, open Database file.

MMQuery_GetFMErrorText

Description

This function returns the Error Text associated with a FileMaker Pro error number.

Return Type

Text

Format

MMQuery_GetFMErrorText (**ErrorNumber**)

Required Parameters

ErrorNumber
The Error Number of the Error Text you want.

Example

Code:

```
MMQuery_GetFMErrorText( 106 )
```

Result:

Returns "Table is missing (106)".

MMQuery_GetLayoutFieldNames

Description

This function returns a list of Field Names from a Layout. The first parameter is the Name of the Layout to return the Field Names from. If you do not specify this parameter, MMQuery will use Get(LayoutName) to get the current Layout to return the Field Names from. By default, MMQuery will check to make sure the Layout Name you specify exists before getting the Field Names. If you already know for sure that the Layout Name exists, you can pass True as the second parameter.

Return Type

Text

Format

MMQuery_GetLayoutFieldNames (**Layout Name** ; **SkipLayoutNameCheck**)

Optional Parameters

Layout Name
The Name of the Layout that contains the Fields. You can specify "" here to have MMQuery use the current Layout.

SkipLayoutNameCheck
If True, MMQuery will not check to make sure the Layout Name you specify actually exists.

Related Items

MMQuery_GetLayoutIDs, MMQuery_GetLayoutNames, MMQuery_LayoutExists

Example

Code:

```
MMQuery_GetLayoutFieldNames( "MyLayout" )
```

Result:

Returns something like:<br/>
AField<br/>
BField<br/>
Related::DField

MMQuery_GetLayoutIDs

Description

This function returns either the ID of a specific Layout or a list of "LayoutName=LayoutID" pairs for all Layouts in the current Database File. If you specify a Layout Name in the first parameter, MMQuery will return the Layout ID of that specific Layout. If you do not specify this parameter, MMQuery will return a list of "LayoutName=LayoutID" pairs for all the Layouts in the current Database File. By default, MMQuery will check to make sure the Layout Name you specify exists before getting the Layout IDs. If you already know for sure that the Layout name exists, you can pass True as the second parameter.

Return Type

Text or Number

Format

MMQuery_GetLayoutIDs (**LayoutName** ; **SkipLayoutNameCheck**)

Optional Parameters

LayoutName
If you specify this parameter, MMQuery will return the ID of the Layout you specify. Otherwise, MMQuery will return all the IDs of all the Layouts.

SkipLayoutNameCheck
If True, MMQuery will not check to make sure the Layout Name you specify actually exists.

Related Items

MMQuery_GetLayoutFieldNames, MMQuery_GetLayoutNames, MMQuery_LayoutExists

Examples

Example 1

Code:

```
MMQuery_GetLayoutIDs
```

Result:

Returns something like:<br/>
ZLayout=1<br/>
YLayout=2

Example 2

Code:

```
MMQuery_GetLayoutIDs( "YLayout" )
```

Result:

Returns 2 if the Layout ID for the "YLayout" Layout in the current Database File is 2.

MMQuery_GetLayoutNames

Description

This function returns a list of Layout Names from the current Database File.

Return Type

Text

Format

MMQuery_GetLayoutNames

Related Items

MMQuery_GetLayoutFieldNames, MMQuery_GetLayoutIDs, MMQuery_LayoutExists

Example

Code:

```
MMQuery_GetLayoutNames
```

Result:

Returns something like:<br/>
ZLayout<br/>
YLayout

MMQuery_GetRecordData

Description

Note: This function is only available in FileMaker Pro 11 and above.

When using the new MMQuery_ExecuteSQLEx function or the modified MMQuery_FindRecords function, you can optionally specify that the results be stored internally in the plug-in. This function is used to retrieve those results. The first parameter is the Row or Record number of the result to look in. The second parameter is the Column or Field number of the result to look in.

Return Type

Varies

Format

MMQuery_GetRecordData (**Row** ; **Column**)

Required Parameters

Row
The Row of Record Data to return. Note: The first Row is at position 0 (zero).

Column
The Column of Record Data to return. Note: The first Column is at position 0 (zero).

Related Items

MMQuery_ExecuteSQLEx, MMQuery_FindRecords, MMQuery_GetRecordDataColumnCount, MMQuery_GetRecordDataRowCount

Example

Code:

```
MMQuery_GetRecordData( 0 ; 4 )
```

Result:

Returns the value of the field in the first record of results in the 4th field.

MMQuery_GetRecordDataColumnCount

Description

Note: This function is only available in FileMaker Pro 11 and above.

This function returns the number of Columns or Fields in the current stored Record Data.

Return Type

Number

Format

MMQuery_GetRecordDataColumnCount

Related Items

MMQuery_GetRecordData, MMQuery_GetRecordDataRowCount

Example

Code:

```
MMQuery_GetRecordDataColumnCount
```

Result:

This would return 4 if the current stored Record Data had 4 columns of data.

MMQuery_GetRecordDataRowCount

Description

Note: This function is only available in FileMaker Pro 11 and above.

This function returns the number of Rows or Records in the current stored Record Data.

Return Type

Number

Format

MMQuery_GetRecordDataRowCount

Related Items

MMQuery_GetRecord, MMQuery_GetRecordDataColumnCount

Example

Code:

```
MMQuery_GetRecordDataRowCount
```

Result:

This would return 10 if the current stored Record Data had 10 rows of data.

MMQuery_GetTOIDs

Description

This function returns either the ID of a specific Table Occurrence or a list of "TOName=TOID" pairs for all Table Occurrences in the Database File. If you specify a TO Name in the first parameter, MMQuery will return the Table Occurrence ID of that specific Table Occurrence. If you do not specify this parameter, MMQuery will return a list of "TOName=TOID" pairs for all the TOs in the Database File. By default, MMQuery will check to make sure the TO Name you specify exists before getting the Table ID. If you already know for sure that the TO Name exists, you can pass True as the second parameter.

Return Type

Text or Number

Format

MMQuery_GetTOIDs (**TOName** ; **SkipTONameCheck**)

Optional Parameters

TOName
If you specify this parameter, MMQuery will return the ID of the Table Occurrence you specify. Specify "" to return the Table Occurrence IDs of all Table Occurrences. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) followed by a period to retrieve all the Table Occurrence IDs in that file (eg. "FileName."). Also in FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specify actually exists.

Related Items

MMQuery_AddTable, MMQuery_GetBaseTableNames, MMQuery_GetTONames, MMQuery_RemoveTable, MMQuery_TOExists

Examples

Example 1

Code:

```
MMQuery_GetTOIDs
```

Result:

Returns something like:<br/>
MyTable=1065089<br/>
AnotherTable=1065092

Example 2

Code:

```
MMQuery_GetTOIDs( "AnotherTable" )
```

Result:

Returns 1065092 if the TO ID for the "AnotherTable" Table in the current Database File is 1065092.

Example 3

Code:

```
MMQuery_GetTOIDs( "AnotherFile." )
```

Result:

In FileMaker Pro 11 and above, use this form to get all the Table Occurrence IDs from a separate, open Database file.

Example 4

Code:

```
MMQuery_GetTOIDs( "AnotherFile.MyTable" )
```

Result:

In FileMaker Pro 11 and above, use this form to get the Table Occurrence ID from a Table Occurrence in a separate, open Database file.

MMQuery_GetTONames

Description

This function returns a list of Table Occurrence Names from the Database File. With FileMaker Pro 11, if the optional FileName parameter is defined, this function will return a list of the Table Occurrence Names from that Database File.

Return Type

Text

Format

MMQuery_GetTONames (**FileName**)

Optional Parameters

FileName
If you specify this parameter, MMQuery will return the Table Occurrence Names from this Database File (if it is open).

Note: This parameter is only available with FileMaker Pro 11 and above.

Related Items

MMQuery_AddTable, MMQuery_GetBaseTableNames, MMQuery_GetTOIDs, MMQuery_RemoveTable, MMQuery_TOExists

Examples

Example 1

Code:

```
MMQuery_GetTONames
```

Result:

Returns something like:<br/>
MyTable<br/>
AnotherTable

Example 2

Code:

```
MMQuery_GetTONames( "AnotherFile" )
```

Result:

In FileMaker Pro 11 and above, use this form to get the Table Occurrence Names from a separate, open Database file.

MMQuery_LayoutExists

Description

This function checks for the Existence of a Layout in the current Database File. The parameter is the Name of the Layout to look for.

Return Type

Number (1=True, 0=False)

Format

MMQuery_LayoutExists (**LayoutName**)

Required Parameters

LayoutName
The Name of the Layout to search for.

Related Items

MMQuery_GetLayoutFieldNames, MMQuery_GetLayoutIDs, MMQuery_GetLayoutNames

Example

Code:

```
MMQuery_LayoutExists( "ZLayout" )
```

Result:

Returns 1 when a Layout Named "ZLayout" exists in the current Database File.

MMQuery_Register

Description

You can use this function to Register the plug-in from a script instead of through the Configuration Dialog. This is useful when the plug-in is being distributed to many computers, allowing you to intall and register the plug-in without having to physically visit each computer. This function also allows you to check if the plug-in is already registered or clear the current registration. The plug-in always requires you to accept the License Agreement to use the plug-in. This is usually done by presenting the License Agreement Dialog, but that can be suppressed by using the special option value "I Accept the License Agreement".

Return Type

Text

Format

MMQuery_Register (**FirstName** ; **LastName** ; **LicenseKey** ; **Option**)

Required Parameters

FirstName
The Registration First Name you specified when you ordered. (See your Receipt.)

LastName
The Registration Last Name you specified when you ordered. (See your Receipt.)

LicenseKey
The License Key from your Receipt.

Optional Parameters

Option
Specify "Dialog" to enter your Registration Information in a dialog.
Specify "Check" to see if the plug-in is already Registered.
Specify "Clear" to remove the Registration.
Specify "I Accept the License Agreement" to automatically accept the License Agreement dialog without showing it to the end user.
Notes: The "Dialog", "Check", and "Clear" options can also be specified as the first parameter. If you have a Developer License, do not use the "I Accept the License Agreement" value here. See your Developer Instructions file for more information.

Examples

Example 1

Code:

```
MMQuery_Register( "My First Name" ; "My Last Name" ; "My License Key" )
```

Result:

Registers the plug-in with the provided registration information (obviously the above is not valid registration information; please see your Receipt).

Example 2

Code:

```
MMQuery_Register( "Dialog" )
```

Result:

Displays a dialog for you to enter your First Name, Last Name, and Registration Number as it appears on your Receipt.

Example 3

Code:

```
MMQuery_Register( "Check" )
```

Result:

Returns "Not Registered." or "Registered to <name> for a <license>."

Example 4

Code:

```
MMQuery_Register( "My Company Name" ; "My Company Name" ; "My Site License Number" ; "I Ac
```

Result:

Registers the plug-in and uses the "I Accept the License Agreement" option to keep the License Agreement dialog from appearing.

MMQuery_RemoveField

Description

This function Removes a Field from a Table. The first parameter is the Name of a Table Occurrence of the Table to Remove the Field from. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout TO and Remove the Field from that TO. The second parameter is the Name of the Field to Remove. By default, MMQuery will check to make sure the TO Name and the Field Name you specify exist before attempting to Remove the Field. If you already know for sure that the TO Name and the Field Name exist, you can pass True as the third and fourth parameters.

Note: In FileMaker Pro 8.5 and above, you must insert a "Pause" Script Step after using this function in order for the Field to be Removed.

Return Type

Text

Format

MMQuery_RemoveField (**TOName** ; **FieldName** ; **SkipTONameCheck** ; **SkipFieldNameCheck**)

Required Parameters

TOName
The Name of the Table Occurrence to Remove the Field from. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldName
The Name of the Field to Remove.

Optional Parameters

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before attempting to Remove the Field.

SkipFieldNameCheck
If True, MMQuery will not check to make sure the Field Name you specified actually exists before attempting to Remove the Field.

Related Items

MMQuery_AddField, MMQuery_FieldExists, MMQuery_GetFieldIDs, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByFind, MMQuery_GetFieldValueByID, MMQuery_SetFieldValueByFind, MMQuery_SetFieldValueByID

Examples

Example 1

Code:

```
MMQuery_RemoveField( "MyTable" ; "MyField" )
```

Result:

Removes the "MyField" Field from the "MyTable" Table.

Example 2

Code:

```
MMQuery_RemoveField( "AnotherFile.MyTable" ; "MyField" )
```

Result:

In FileMaker Pro 11 and above, use this form to Remove a Field from a Table in a separate, open Database file.

MMQuery_RemoveRecordByFind

Description

This function Removes one or more Records based on a Find Criteria. The first parameter is the Name of a Table Occurrence for the Table to Remove the Record from. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout TO to Remove the Record from. The second parameter is a return- or paragraph mark-separated list of "FieldName=FieldValue" pairs that describes the Criteria for Finding the Record or Records to Remove. See the "Finding What You Want" section in the MMQuery_FindRecords function for help in defining your Find Criteria. By default, MMQuery will only Remove the first Record it finds. If you want to Remove all Records that match the Find Criteria, specify True as the third parameter. By default, MMQuery will check to make sure the TO Name and the Field Names you specify exist before attempting to Remove the Record. If you already know for sure that the TO Name and the Field Names exist, you can pass True as the fourth and fifth parameters. By default, MMQuery will double-check to make sure the Record or Records were actually removed. If you would rather the plug-in skip this step, specify True as the sixth parameter. Finally, if you want the Find to be Case Sensitive, you can specify True as the last parameter.

Return Type

Text

Format

MMQuery_RemoveRecordByFind (**TOName** ; **FindCriteria** ; **RemoveAllMatches** ; **SkipTONameCheck** ; **SkipFieldNameChecks** ; **SkipConfirmRemoval** ; **CaseSensitive**)

Required Parameters

TOName
The Name of the Table Occurrence that contains the Record you want to Remove. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FindCriteria
A return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the Criteria for Finding the Record to Remove. See the "Finding What You Want" section of the MMQuery_FindRecords function for help in setting up the Find Criteria.

Optional Parameters

RemoveAllMatches
If True, MMQuery will Remove all Records that match the Find Criteria. Default is False.

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before Removing the Record.

SkipFieldNameChecks
If True, MMQuery will not check to make sure the Field Names you specified actually exist before Removing the Record.

SkipConfirmRemoval
If True, MMQuery will not double-check to make sure the Record was Removed.

CaseSensitive
If True, MMQuery will preserve the Case in the Field Values of the Find Criteria. (In other words, if

CaseSensitive is True, "Green" would not match "green".)

Related Items

MMQuery_AddRecord, MMQuery_FindRecords, MMQuery_RemoveRecordByID, MMQuery_UpdateRecordByFind, MMQuery_UpdateRecordByID

Examples

Example 1

Code:

```
MMQuery_RemoveRecordByFind( "MyTable" ; "AField=some value" )
```

Result:

This would Remove the first Record in the "MyTable" Table where the "AField" Field had the value "some value".

Example 2

Code:

```
MMQuery_RemoveRecordByFind( "" ; "::Begins-With::LastName=J" ; True )
```

Result:

This would Remove all Records in the current Table where the Values of the "LastName" field started with a "J".

Example 3

Code:

```
MMQuery_RemoveRecordByFind( "AnotherFile.MyTable" ; "AField=some value" )
```

Result:

In FileMaker Pro 11 and above, use this form to Remove a Record from a Table Occurrence in a separate, open Database file.

MMQuery_RemoveRecordByID

Description

This function Removes a Record from a Table using its Internal FileMaker Record ID. The first parameter is the Name of a Table Occurrence of the Table to Remove the Record from. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout TO to Remove the Record from. The second parameter is the Internal FileMaker Record ID of the Record you want to Remove. By default, MMQuery will check to make sure the TO Name and the Record ID you specify actually exist before attempting to Remove the Record. If you already know for sure that the TO Name and the Record ID exist, you can pass True as the third and fourth parameters. By default, MMQuery will double-check to make sure the Record was actually Removed. If you would rather the plug-in skip this step, specify True as the fifth parameter.

Return Type

Text

Format

MMQuery_RemoveRecordByID (**TOName** ; **FMRecID** ; **SkipTONameCheck** ; **SkipFMRecIDCheck** ; **SkipConfirmRemoval**)

Required Parameters

TOName
The Name of the Table Occurrence that contains the Record you want to Remove. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FMRecID
The Internal FileMaker Record ID of the Record to Remove.

Optional Parameters

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before Removing the Record.

SkipFMRecIDCheck
If True, MMQuery will not check to make sure the Record ID you specified actually exists before Removing the Record.

SkipConfirmRemoval
If True, MMQuery will not double-check to make sure the Record was Removed.

Related Items

MMQuery_AddRecord, MMQuery_FindRecords, MMQuery_RemoveRecordByFind, MMQuery_UpdateRecordByFind, MMQuery_UpdateRecordByID

Examples

Example 1

Code:

```
MMQuery_RemoveRecordByID( "MyTable" ; 2 )
```

Result:

This would Remove the Record with the Record ID 2 from the "MyTable" Table.

Example 2

Code:

```
MMQuery_RemoveRecordByID( "AnotherFile.MyTable" ; 4 )
```

Result:

In FileMaker Pro 11 and above, use this form to Remove a Record from a Table Occurrence in a separate, open Database file.

MMQuery_RemoveTable

Description

This function Removes the Base Table corresponding to the Table Occurrence Name you specify. The first parameter is the Name of a Table Occurrence for the Table to Remove.

Return Type

Text

Format

MMQuery_RemoveTable (**TOName**)

Required Parameters

TOName
The Name of the Table Occurrence corresponding to the Base Table you want to Remove. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

Related Items

MMQuery_AddTable, MMQuery_GetBaseTableNames, MMQuery_GetTOIDs, MMQuery_GetTONames, MMQuery_TOExists

Examples

Example 1

Code:

```
MMQuery_RemoveTable( "MyOldTable" )
```

Result:

Removes the "MyOldTable" Table from the current Database File.

Example 2

Code:

```
MMQuery_RemoveTable( "AnotherFile.MyOldTable" )
```

Result:

In FileMaker Pro 11 and above, use this form to Remove a Base Table with the Table Occurrence specified in a separate, open Database file.

MMQuery_SetFieldValueByFind

Description

This function Sets the Value of a single Field using Find Criteria. The first parameter is the Name of a Table Occurrence of the Table to Set the Field Value in. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout TO to Set the Field Value in. The second parameter is the Name of the Field to set. The third parameter is the new Value for the Field. The fourth parameter is a return- or paragraph mark-separated list of "FieldName=FieldValue" pairs that describes the Criteria for Finding the Record to Set the Field Value in. See the "Finding What You Want" section in the MMQuery_FindRecords function for help in defining your Find Criteria. By default, MMQuery will check to make sure the TO Name and all the Field Names you specify exist before attempting to Set the Field Value. If you already know for sure that the TO Name and the Field Names exist, you can pass True as the fifth and sixth parameters. Finally, if you want the Find to be Case Sensitive, you can specify True as the last parameter.

Return Type

Text

Format

MMQuery_SetFieldValueByFind (**TOName** ; **FieldName** ; **FieldValue** ; **FindCriteria** ; **SkipTONameCheck** ; **SkipFieldNameChecks** ; **CaseSensitive**)

Required Parameters

TOName
The Name of the Table Occurrence to Set the Field Value in. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldName
The Name of the Field to Set.

FieldValue
The new Value for the Field.

FindCriteria
A return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the Criteria for Finding the Record to Set the Field Value in. See the "Finding What You Want" section of the MMQuery_FindRecords function for help in setting up the Find Criteria.

Optional Parameters

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before Setting the Field Value.

SkipFieldNameChecks
If True, MMQuery will not check to make sure all the Field Names in the variaous parameters actually exist before Setting the Field Value.

CaseSensitive
If True, MMQuery will preserve the Case in the Field Values of the Find Criteria. (In other words, if CaseSensitive is True, "Green" would not match "green".)

Related Items

MMQuery_AddField, MMQuery_FieldExists, MMQuery_FindRecords, MMQuery_GetFieldIDs, MMQuery_GetFieldNames, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByFind, MMQuery_GetFieldValueByID, MMQuery_RemoveField, MMQuery_SetFieldValueByID

Examples

Example 1

Code:

```
MMQuery_SetFieldValueByFind( "MyTable" ; "BField" ; "A New Value" ; "AField=some value" )
```

Result:

This would Set the Value of the "BField" Field to "A New Value" in the "MyTable" Table for the first Record where the "AField" Field has the Value "some value".

Example 2

Code:

```
MMQuery_SetFieldValueByFind( "" ; "LastName" ; "Jones" ; "FirstName=Ann" )
```

Result:

This would Set the Value of the "LastName" Field to "Jones" in the current Table for the first Record where the "FirstName" Field has the Value "Ann".

Example 3

Code:

```
MMQuery_SetFieldValueByFind( "AnotherFile.MyTable" ; "BField" ; "A New Value" ; "AField=so
```

Result:

In FileMaker Pro 11 and above, use this form to Set a Field Value in a Table Occurrence in a separate, open Database file.

MMQuery_SetFieldValueByID

Description

This function Sets the Value of a single Field in a Record with a specific ID. The first parameter is the Name of a Table Occurrence of the Table to Set The Field Value in. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout TO to Set the Field Value in. The second parameter is the Name of the Field to Set. The third parameter is the new Value for the Field. The fourth parameter is the Internal FileMaker Record ID of the Record to Set the Field Value in. If you do not specify this parameter, MMQuery will use Get(RecordID) to get the Record ID. By default, MMQuery will check to make sure the TO Name, the Field Name, and the Record ID you specify exist before attempting to Set the Field Value. If you already know for sure that the TO Name, the Field Name, and the Record ID exist, you can pass True as the fifth, sixth, and seventh parameters.

Return Type

Text

Format

MMQuery_SetFieldValueByID (**TOName** ; **FieldName** ; **FieldValue** ; **FMRecID** ; **SkipTONameCheck** ; **SkipFieldNameCheck** ; **SkipFMRecIDCheck**)

Required Parameters

TOName
The Name of a Table Occurrence of the Table to Set the Field Value in. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FieldName
The Name of the Field to Set.

FieldValue
The new Value for the Field.

Optional Parameters

FMRecID
The Internal FileMaker Record ID of the Record to Set the Field Value in.

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before Setting the Field Value.

SkipFieldNameCheck
If True, MMQuery will not check to make sure the Field Name you specified actually exists before Setting the Field Value.

SkipFMRecIDCheck
If True, MMQuery will not check to make sure the Record ID you specified actually exists before Setting the Field Value.

Related Items

MMQuery_AddField, MMQuery_FieldExists, MMQuery_GetFieldIDs, MMQuery_GetFieldNames, MMQuery_GetFieldTypes, MMQuery_GetFieldValueByFind, MMQuery_GetFieldValueByID, MMQuery_RemoveField, MMQuery_SetFieldValueByFind

Examples

Example 1

Code:

```
MMQuery_SetFieldValueByID( "MyTable" ; "AField" ; "some new value" ; 4  )
```

Result:

This would Set the Value of the "AField" Field in the "MyTable" Table" with the Record with ID 4 to the Value "some new value".

Example 2

Code:

```
MMQuery_SetFieldValueByID( "" ; "FirstName" ; "Maryanne" )
```

Result:

This would Set the Value of the "FirstName" Field of the current Record in the current Table to the Value "Maryanne".

Example 3

Code:

```
MMQuery_SetFieldValueByID( "AnotherFile.MyTable" ; "AField" ; "some new value" ; 4 )
```

Result:

In FileMaker Pro 11 and above, use this form to Set a Field Value in a Table in a separate, open Database file.

MMQuery_TOExists

Description

This function checks for the Existence of a Table Occurrence in the current Database File. The parameter is the Name of the Table Occurrence to look for.

Return Type

Number (1=True, 0=False)

Format

MMQuery_TOExists (**TOName**)

Required Parameters

TOName
The Name of the Table Occurrence to search for. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

Related Items

MMQuery_AddTable, MMQuery_GetBaseTableNames, MMQuery_GetTOIDs, MMQuery_GetTONames, MMQuery_RemoveTable

Examples

Example 1

Code:

```
MMQuery_TableExists( "MyTable" )
```

Result:

Returns 1 when a Table Named "MyTable" exists in the current Database File.

Example 2

Code:

```
MMQuery_TOExists( "AnotherFile.MyTable" )
```

Result:

In FileMaker Pro 11 and above, use this form to check for a Table Occurrence in a separate, open Database file.

MMQuery_UpdateRecordByFind

Description

This function Updates one or more Records based on Find Criteria. The first parameter is the Name of a Table Occurrence of the Table to Update the Record in. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout TO to Update the Record in. The second parameter is a return- or paragraph mark-separated list of "FieldName="FieldValue" pairs describing the Criteria for Finding the Record or Records to Update. See the "Finding What You Want" section of the MMQuery_FindRecords function for help in defining your Find Criteria. The third parameter is a return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the Field Values to Update the Record with. If a Field Value needs a return in it, you can use "<CR>" to indicate a return. By default, MMQuery will only Update the first Record it finds. If you want to Update all Records that match the Find Criteria, specify True as the fourth parameter. By default, MMQuery will check to make sure the TO Name and all the Field Names exist before attempting to Update the Record. If you already know for sure that the TO Name and the Field Names exist, you can pass True as the fifth and sixth parameters. Finally, if you want the Find to be Case Sensitive, you can specify True as the last parameter.

Return Type

Text

Format

MMQuery_UpdateRecordByFind (**TOName** ; **FindCriteria** ; **FieldValues** ; **UpdateAllMatches** ; **SkipTONameCheck** ; **SkipFieldNameChecks** ; **CaseSensitive**)

Required Parameters

TOName
The Name of a Table Occurrence of the Table that contains the Record you want to Update. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FindCriteria
A return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the Criteria for Finding the Record to Update. See the "Finding What You Want" section of the MMQuery_FindRecords function for help in setting up the Find Criteria.

FieldValues
A return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the new Field Values to Update the Record with. Use "<CR>" to indicate a return or new line in a Field Value.

Optional Parameters

UpdateAllMatches
If True, MMQuery will Update all Records that match the Find Criteria. Default is False.

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before Updating the Record.

SkipFieldNameChecks
If True, MMQuery will not check to make sure all the Field Names you specified in the various parameters actually exist before Updating the Record.

CaseSensitive
If True, MMQuery will preserve the Case in the Field Values of the Find Criteria. (In other words, if CaseSensitive is True, "Green" would not match "green".)

Related Items

MMQuery_AddRecord, MMQuery_FindRecords, MMQuery_RemoveRecordByFind, MMQuery_RemoveRecordByID, MMQuery_UpdateRecordByID

Examples

Example 1

Code:

```
MMQuery_UpdateRecordByFind( "MyTable" ; "AField=some value" ; "BField=some new value¶CFiel
```

Result:

Updates the first Record in the "MyTable" Table where the "AField" Field has the Value "some value", setting the "BField" Field to "some new value" and the "CField" Field to "some other value".

Example 2

Code:

```
MMQuery_UpdateRecordByFind( "" ; "State=Texas" ; "Region=South" ; True )
```

Result:

Updates all Records in the current Table where the "State" Field has the value "Texas", setting the "Region" Field to the Value "South".

Example 3

Code:

```
MMQuery_UpdateRecordByFind( "AnotherFile.MyTable" ; "AField=some value" ; "BField=some new
```

Result:

In FileMaker Pro 11 and above, use this form to Update a Record in a Table Occurrence in a separate, open Database file.

MMQuery_UpdateRecordByID

Description

This function Updates a Record in a Table Occurrence using its Internal FileMaker Record ID. The first parameter is the Name of a Table Occurrence of the Table to Update the Record in. If you do not specify this parameter, MMQuery will use Get(LayoutTableName) to get the Current Layout Table to Update the Record in. The second parameter is the Internal FileMaker Record ID of the Record you want to Update. The third parameter is a return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the Field Values to Update the Record with. If a Field Value needs a return in it, you can use "<CR>" to indicate a return. By default, MMQuery will check to make sure the TO Name, the Record ID, and the Field Names you specify exist before attempting to Update the Record. If you already know for sure that the TO Name, the Record ID, and the Field Names exist, you can pass True as the fourth, fifth, and sixth parameters.

Return Type

Text

Format

MMQuery_UpdateRecordByID (**TOName** ; **FMRecID** ; **FieldValues** ; **SkipTONameCheck** ; **SkipFMRecIDCheck** ; **SkipFieldNameChecks**)

Required Parameters

TOName
The Name of the Table Occurrence that contains the Record you want to Update. Specify "" to use the Current Layout TO. In FileMaker Pro 11 and above, you can optionally specify the Database Filename (without the ".fp7" extension) containing the TO in the format "FileName.TOName".

FMRecID
The Internal FileMaker Record ID of the Record to Update.

FieldValues
A return- or paragraph mark-separated list of "FieldName=FieldValue" pairs describing the new Field Values to Update the Record with. Use "<CR>" to indicate a return or new line in a Field Value.

Optional Parameters

SkipTONameCheck
If True, MMQuery will not check to make sure the TO Name you specified actually exists before Updating the Record.

SkipFMRecIDCheck
If True, MMQuery will not check to make sure the Record ID you specified actually exists before Updating the Record.

SkipFieldNameChecks
If True, MMQuery will not check to make sure the Field Names you specified actually exist before Updating the Record.

Related Items

MMQuery_AddRecord, MMQuery_FindRecords, MMQuery_RemoveRecordByFind, MMQuery_RemoveRecordByID, MMQuery_UpdateRecordByFind

Examples

Example 1

Code:

```
MMQuery_UpdateRecordByID( "MyTable" ; 2 ; "BField=some new value¶CField=some other value"
```

Result:

Updates the Record with the ID 2 in the "MyTable" Table, setting the "BField" Field to "some new value" and the "CField" Field to "some other value".

Example 2

Code:

```
MMQuery_UpdateRecordByID( "" ; 4 ; "Description=some<CR>value<CR>with<CR>multiple<CR>lines
```

Result:

Updates the Record with ID 4 in the current Table, setting the "Description" field with a multi-line value.

Example 3

Code:

```
MMQuery_UpdateRecordByID( "AnotherFile.MyTable" ; 4 ; "BField=some new value" )
```

Result:

In FileMaker Pro 11 and above, use this form to Update a Record in a Table Occurrence in a separate, open Database file.

MMQuery_Version

Description

This function returns the current version of MMQuery. This function is useful for testing whether or not the plug-in is installed and enabled. If you call this function and a question mark ("?") is returned, then the plug-in is either not installed or not enabled.

Return Type

Text

Format

MMQuery_Version

Related Items

MMQuery_VersionAutoUpdate

Example

Code:

```
MMQuery_Version
```

Result:

Returns the MMQuery version like "MMQuery v.1.1.1".

MMQuery_VersionAutoUpdate

Description

This function returns an Auto Update friendly Version number of MMQuery. The format of this version number is always exactly 8 digits long. The first two digits represent the major version of the plug-in (zero-filled). The third and fourth digits represent the minor version of the plug-in (zero-filled). The fifth and sixth digits represent the update portion of the version (zero-filled). The final two digits represent a special build number or a beta version number and will usually be zeros.

As an example, for MMQuery 1.1.1, the major version is 1, the minor version is 1, the update number is 1, and there is no special build or beta version defined. So, the resulting Auto Update friendly version number would be 01010100.

Return Type

Number

Format

MMQuery_VersionAutoUpdate

Related Items

MMQuery_Version

Example

Code:

```
MMQuery_VersionAutoUpdate
```

Result:

Returns 01010100 for MMQuery version 1.1.1.