Troi Dialog Plug-in 6.5
for FileMaker Pro 15
USER GUIDE

May 2016

Troi Automatisering
Boliviastraat 11
2408 MX Alphen a/d Rijn
The Netherlands

You can also visit the Troi web site at: <http://www.troi.com/> for additional information.

Table of Contents

Installing plug-ins

Starting with FileMaker Pro 12 a plug-in can be installed directly from a container field. Please see the **EasyInstallTroiPlugins.fmp12** example file to install plug-ins with FileMaker Pro 12, 13, 14 or 15.

The instructions below show instructions for FileMaker Pro 11.

For Mac OS X:

- Quit FileMaker Pro.
- Put the file “Troi_Dialog.fmplugin” from the folder “Mac OS Plug-in” into the “ Extensions” folder in the FileMaker Pro application folder.
- If you have installed previous versions of this plug-in, you are asked: “An older item named “Troi_Dialog.fmplugin” already exists in this location. Do you want to replace it with the one you’re moving?’. Press the OK button.
- Start FileMaker Pro. The first time the Troi Dialog Plug-in is used it will display a dialog box, indicating that it is loading and showing the registration status.

For Windows:

- Quit FileMaker Pro.
- Put the file "Troi_Dialog.fmx" from the directory "Windows Plug-in" into the "Extensions" subdirectory in the FileMaker Pro directory.
- If you have installed previous versions of this plug-in, you are asked: “This folder already contains a file called 'Troi_Dialog.fmx'. Would you like to replace the existing file with this one?’. Press the Yes button.
- Start FileMaker Pro. The Troi Dialog Plug-in will display a dialog box, indicating that it is loading and showing the registration status.

TIP You can check which plug-ins you have loaded by going to the plug-in preferences: Choose **Preferences** from the menu, and then choose **Plug-ins**.

You can now open the example file "All Dialog Examples" to see how to use the plug-in's functions. There is also a Function overview available.

If You Have Problems

This user guide tries to give you all the information necessary to use this plug-in. So if you have a problem please read this user guide first. Also you might visit our support web page:

<http://www.troi.com/support/>

This page contains FAQ's (Frequently Asked Questions), help on registration and much more. If that doesn't help, you can get free support by email. Send your questions to **support@troi.com** with a full explanation of the problem. Also give as much relevant information (version of the plug-in, which platform, version of the operating system, version of FileMaker Pro) as possible.

If you find any mistakes in this manual or have a suggestion please let us know. We appreciate your feedback!

TIP You can get more information on returned error codes from the OSErrrs database on our web site: <http://www.troi.com/software/oserrrs.html>. This free FileMaker database lists all error codes for Windows and Mac OS X!

What can this plug-in do?

The Troi Dialog Plug-in is a very powerful tool for adding dynamic dialog functions to FileMaker Pro. With it you can show several types of dialogs from a FileMaker Script, where the dialog text and the buttons are generated dynamically. All from within FileMaker you can:

- show sophisticated (input) dialogs
- show password dialogs
- show flash dialogs
- show list dialogs
- show big input dialogs.
- show a slideshow of images and movies

Software requirements

System requirements for Mac OS X

Mac OS X 10.6.8 (Snow Leopard), Mac OS X 10.7 (Lion), OS X 10.8 (Mountain Lion), OS X 10.9 (Mavericks), OS X 10.10 (Yosemite), OS X10.11 (El Capitan).

System requirements for Windows

Windows 7 on Intel-compatible computer 1 GHz or faster
Windows 8 or Windows 8.1
Windows 10

FileMaker requirements

FileMaker Pro 12 or FileMaker ProAdvanced 12 or higher.
FileMaker Pro 13 or FileMaker Pro Advanced 13 or higher.
FileMaker Pro 14 or FileMaker Pro Advanced 14 or higher.
FileMaker Pro 15 or FileMaker Pro Advanced 15 or higher.

NOTE We have successfully tested Troi Dialog Plug-in with FileMaker Pro 11, but we no longer provide active support for this version. Troi Dialog plug-in will also probably run with FileMaker 7 to 10, but we have not tested this and we no longer provide support for this..

FileMaker Server

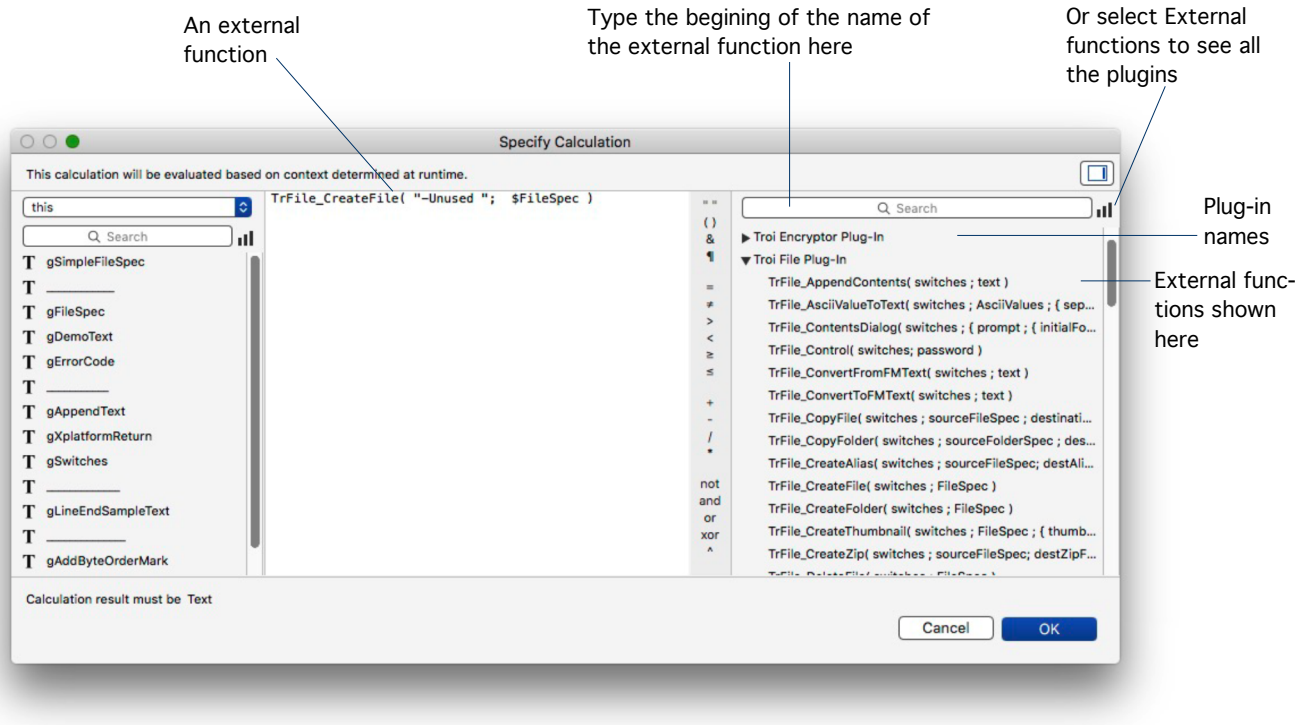
You can also use FileMaker Server 12 to 15 to serve databases that use functions of the Troi Dialog Plug-in (client-side). You need to have the plug-in installed at the **clients** that use these functions.

Use the **EasyInstallTroiPlugins.fmp12** example file to install plug-ins with FileMaker Pro 12 to 15

Getting started

Using external functions

Troi Dialog Plug-in adds new functions to the standard functions that are available in FileMaker Pro. The functions added by a plug-in are called external functions. You can see those extra functions for all plug-ins at the top right of the Specify Calculation Box:



You use special syntax with external functions: FunctionName(parameter1 ; parameter 2) where FunctionName is the name of an external function. A function can have zero or more parameters. Each parameter is separated by a semi-colon. Plug-ins don't work directly after installation. To access a plug-in function, you need to add the calls to the function in a calculation, for example in a Set Field or Set Variable script step in ScriptMaker.

Where to add the External Functions?

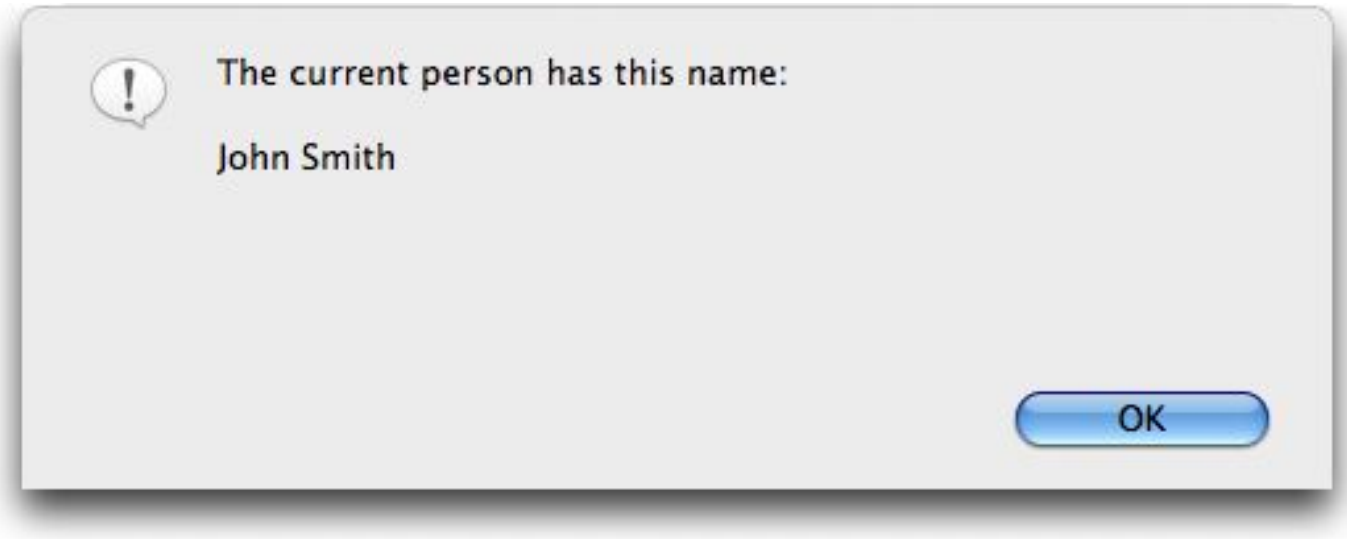
External functions for this plug-in are intended to be used in a script step using a calculation. For most functions of this plug-in it makes no sense to add them to a define field calculation, as the functions will have side effects.

Simple example

Say you have a database Persons.fmp12, with a text field called NameTotal. You want to display a dialog where you display this name from the current record. In ScriptMaker create a script "Display current name". Add the following script step to this script:

```
Set Field[myResultField, Dial_Dialog( "-NoteIcon" ;
          "The current person has this name: ¶¶" & NameTotal ; "OK")]
```

This will show you a dialog with the text and an OK button, similar to this:



Note that function names, like `Dial_Dialog` are no longer case sensitive.

Using Icons

You can specify which icon to add to a dialog. This can be one of the standard icons or a custom icon. See below for the standard icons which can be shown:

Mac	Win	
		stop icon
		caution icon
		note icon

Summary of functions

The Troi Dialog Plug-in adds the following functions:

function name	short description
Dial_BigInputDialog	Displays an input dialog box, in which the user can enter a long text.
Dial_DelayTicks	Waits a specified period of time.
Dial_Dialog	Displays a dialog box, this function has a lot of extra possibilities.
Dial_FlashDialog	Displays a flash dialog box for a specified period of time.
Dial_GetButton	Returns the number of the button clicked in the last dialog shown.
Dial_GetCurrentTimestamp	Returns the current timestamp with extra precision, in milliseconds.
Dial_GetInput	Returns the contents of one of the input fields.
Dial_GetPopup	Returns the current popup list stored in the plug-in.
Dial_IconControl	Sets the custom icon.
Dial_InputDialog	Shows an input dialog, with up to 15 input, password, popup or checkbox fields.
Dial_ListDialog	Displays a list dialog box, from which the user can choose an item.
Dial_PresentImage	Displays one or more images or movies full screen or as a slideshow.
Dial_ProgressBar	Displays a progress bar dialog box.
Dial_SetDialogTitle	Sets the default Dialog Title for the title bar of dialogs.
Dial_SetInput	Sets the text to be pre-entered in the input fields of the next input dialog.
Dial_SetLabels	Sets the label text to be displayed next to the input fields of the next input dialog.
Dial_SetListInput	Preselect an item from the items in a list dialog.
Dial_SetPopup	Sets the values to be used for the popup list.
Dial_SetPosition	Sets the position on the screen of dialogs to be shown.
Dial_SetPreference	Sets preferences.
Dial_Version	Use this function to see which version of the plug-in is loaded. This function is also used to register the plug-in.
Dial_VersionAutoUpdate	Returns a version number for the AutoUpdate function of FileMaker Server.

Using the Dial_ProgressBar function

The Troi Dialog Plug-in adds the Dial_ProgressBar function, which manages the display of a progress bar dialog box. With this function you can indicate to a user that a script is doing a lengthy operation, and provide feedback on how long this operation will last. Below you see an example progress bar:

The Dial_ProgressBar function should be implemented in 3 phases:

1- Showing the progress bar: make the progress bar visible.
2- Updating the progress bar: increases the progress bar (multiple times, usually in a loop).
3- Removing the progress bar: the progress bar dialog is removed.

Phase 1: Showing the progress bar

The purpose of this phase is to tell the plug-in to start showing the progress bar and to specify the maximum value the progress bar should go to and to give an initial text. Here is the syntax for this phase:

```
Set Field [result, Dial_ProgressBar("-Unused" ; "show" ;"maxval"; "textToShow" )]
```

The parameter `maxval` should be the result of a calculation of the number of steps. This can be for example based on the number of found records on which the the script is working. Use the Status(CurrentFoundCount) function for this.
An example ScriptMaker Script step can be:

```
Set Field [ result, Dial_ProgressBar("-Unused" ; "show" ;
                    Status(CurrentFoundCount) ; "Starting..." )]
```

NOTE If you specify a zero as maxval, an indefinite progress bar is shown (barber pole).

Phase 2: Updating the progress bar

This phase usually occurs in a loop. When FileMaker has performed a number of actions, like for example processed 10 records, you need to give feedback to the user by updating the display of the progress bar.
Here is the syntax for this:

```
Set Field [ result, Dial_ProgressBar("-Unused" ; "incr" ;
                    increase value; newtext )]
```

The parameter `increase value` should be the result of a calculation of the number of steps that have been handled. You can also give an optional new text to be shown.

So an example ScriptMaker Script step can be:

```
Set Field [ result, Dial_ProgressBar("-Unused" ; "incr" ;
                    10 ; "Still working..." )]
```

Phase 3: Removing the progress bar

If you no longer want to show the progress bar dialog you need to remove the progress bar. This step usually occurs just after the end of the loop. This phase has only "stop" as parameter:

```
Set Field [ result, Dial_ProgressBar("-Unused" ; "stop" ; )]
```

Example script with progress bar

We assume that in your FileMaker file the following fields are defined:

gNumberOfSteps	Global, number
gResult	Global, text

Below you see an example ScriptMaker script which combines the 3 phases:

```
# Calculate the number of steps
Set Field [gNumberOfSteps, Status(CurrentFoundCount)]
# Make the progress bar visible with the initial text
# the 2nd parameter indicates the total number of steps:
Set Field [gResult, Dial_ProgressBar("-Unused" ;
                          "show" ; gNumberOfSteps ;
                          "Showing a simple progress bar...¶¶Please wait.")]
Go to Record[First]
Loop
  # Increase the progress bar by 1
  Set Field [gResult, Dial_ProgressBar("-Unused" ; "incr" ; 1 ; ) ]
  ...
  You should do your time consuming stuff here!
  ...
  Go to Record[Next, Exit after last]
End Loop
# Remove the progress bar
Set Field [gResult, Dial_ProgressBar("-Unused" ; "stop" ;)]
```

Function Reference

Dial_BigInputDialog

Syntax Dial_BigInputDialog(switches ; prompt ; button1 ; button2 ; button3 ; button4 ; initialText)

Displays an input dialog box, in which the user can enter a long text (up to the limit of 64000 characters).

Parameters

switches this determines the behaviour of the dialog and which information is returned.
prompt the text of the dialog
button1..4 the text of the 1st to the 4th button (from the right to left)
initialText the initial text
If you specify an empty button text, no button will be displayed.

Switches can be empty or you can add one or more of these switches:
-ReturnButtonText return the *text* of the button, instead of the number
-SelectAllText select the initial text when the dialog is first shown
-MaxChars=x maximum number of characters allowed. If more characters are typed the dialog beeps
-MinChars=y the minimum number of characters before buttons that require text will be enabled

You can also add one of these switches:
-DefaultButton1 -DefaultButton2 -DefaultButton3 -DefaultButton4
This indicates which button will be selected when the user presses the ENTER key on the Numeric Keypad.

You can also add one or more of these switches:
-Button1NeedsText -Button2NeedsText -Button3NeedsText -Button4NeedsText
If you add this switch the button is disabled until there is (enough) text.

You can also add one of these switches:
-StopIcon shows a stop icon (indicating this is something severe which the user needs to address)
-CautionIcon shows a caution icon (indicating this dialog warns the user)
-NoteIcon shows a note icon (indicating this dialog gives the user information)
-CustomIcon shows a custom icon. This icon needs to be set before with the Dial_IconControl function.

You can also add one or more of these switches:
-IconSize=48 display the icon at 48x48 pixels
-StopOnESC the user can press the ESC key to leave the dialog. The dialog returns with button 0.
-Width=x makes the size of the dialog x pixels wide
-Height=y makes the size of the dialog y pixels high

Returned result

The number of the button that was clicked followed by the text typed. By adding switches this function can also return the text of the button.

The plug-in can also return an error code:
$$-92 ddpLenErr, when the minimum number of characters allowed is bigger than the maximum number of characters.
$$-108 memFullErr, ran out of memory
$$-207 notEnoughBufferSpace, the result is too big.

Other error codes may be returned.

Special considerations

See Dial_SetDialogTitle if you want to change the title of the dialog.

If you type the RETURN key (on Windows labelled as ENTER) on the main keyboard, a return character is entered in the text. You can exit this dialog with the ENTER key on the numeric keypad (normally at the right of the keyboard). The

default button will be returned.

You can set the maximum number of characters a user can type in with the "-MaxChars" switch. For example "-MaxChars=100 will beep when the maximum number of characters is reached. If you don't set this limit the maximum will be 64000 characters.
The -SelectAllText switch will make the initial text selected when the dialog is first shown.

You can also use the keyboard shortcuts: copy (command-C), paste (command-V), cut (command-X) and select all (command-A).

Example usage

```
Set Field [ result, Dial_BigInputDialog(
"-noteIcon -stopOnESC" ; "Please type in your text:" ; "OK" ; "Cancel" ; "Help" ; "Stop" ;
"your story here!¶this on a new line...")  ]
```

This will show an input dialog, with a large text area to type in. The icon is a note icon. The text area has the text "your story here!" and "this on a new line..." already filled in.

Example 2

This is an example for a (simple) dictionary. We assume that in your FileMaker file the following text fields are defined: keywordText and definitionText. Finally a global text field gResult. keywordText should contain keywords and definitionText should contain definitions for that keyword. To make an editing dialog for this dictionary add the following script steps:

```
Set Field [ gResult, Dial_SetDialogTitle( -Unused" ; "Your titletext" ) ]
Set Field [ gResult, Dial_BigInputDialog( "-NoteIcon -Button1NeedsText -DefaultButton1 -dialogName=My Dictionary." ;
     "Please improve this definition of \"" & this::keywordText & "\":" ;
     "OK" ; "Cancel" ; "" ; "" ; "" & this::definitionText ]
Set Field [ definitionText, Dial_GetInput( "-text1") ]
```

This will show a text for a definition taken from the field definitionText. The keyword will be shown between quotes. Note the use of the \" to add the double quote to the calculation.

Dial_DelayTicks

Syntax Dial_DelayTicks(switches ; ticks)

Waits a specified period of time.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
ticks	the time the function waits, in 1/60th of a second

Returned result

This function always returns 0.

Special considerations

A maximum time of 2 minutes can be specified.

On Mac OS X there is a problem where FileMaker does not update the screen after a Dial_DelayTicks step, even if you do a different Set Field step. To workaround this please add this step before the Dial_DelayTicks step: Pause/Resume Script [0:00:00]

Example usage

Set Field[gButtonNr, Dial_DelayTicks("-Unused" ; 180)] will wait for 180 ticks = 3 seconds.

Dial_Dialog

Syntax Dial_Dialog(switches ; prompt ; button1 ; button2 ; button3 ; button4)

Displays a dialog box, from which the user can choose a button.

Parameters

switches	this determines the behaviour of the dialog and which information is returned
prompt	the text of the dialog
button1..4	the text of the 1st to the 4th button (from the right to left)

If you specify an empty button text, no button will be displayed at that place.

Switches can be empty or one of these:

-NoteIcon	shows a note icon (indicating this dialog gives the user information)
-CautionIcon	shows a caution icon (indicating this dialog warns the user)
-StopIcon	shows a stop icon (indicating this is something severe which the user needs to address)
-CustomIcon	shows a custom icon. This icon needs to be set before with the Dial_IconControl function

You can also add this switches:

-IconSize=48	display the icon at 48x48 pixels
-StopOnESC	if you add this switch the user can press the ESC key to leave the dialog. The dialog returns with button 0.

Returned result

The number of the button that was clicked.

Special considerations

See Dial_SetDialogTitle if you want to change the title of the dialog.
See also the functions Dial_GetInput" and Dial_GetButton for easy parsing of the result.
This function does no longer implement the obsolete switches "input", "password" ; "userpassword". When you use one of these switches the function returns the error -4243 = kErrNotImplemented. Please use the more versatile Dial_InputDialog function instead.

Example usage

Dial_Dialog("-NoteIcon " ; "Hi,¶¶Hello World" ; "OK" ; "Cancel" ;)

This shows a dialog with a note icon. If the user chooses the OK button "1" will be returned.If the user chooses the Cancel button "2" will be returned.

Example 2

We assume that in your FileMaker file the following fields are defined:

Name	Text
result	Global, text

The Name field contains names of persons. Add the following script step:

Set Field [result, Dial_Dialog("-CustomIcon" ; "Hi,¶The name is\"" & Name & "\"."; "OK";)]

This shows a dialog with a custom icon and a OK button. The text of the field name is added, so you get something like:

Hi,
The name is "Peter Baanen".

Note the use of the \" to add the double quote to the calculation. The custom icon needs to be set in an earlier script step with the Dial_IconControl function.

Dial_FlashDialog

Syntax Dial_FlashDialog(switches ; ticks ; text)

Displays a flash dialog box for a specified period of time.

Parameters

switches	(optional) this changes the behaviour of the dialog
ticks	The time this dialog must be visible, in 1/60th of a second.
text	Specifies the text you want to display

Switches can be empty or one of these:

-NoteIcon	shows a note icon (indicating this dialog gives the user information)
-CautionIcon	shows a caution icon (indicating this dialog warns the user)
-StopIcon	shows a stop icon (indicating this is something severe which the user needs to address)
-CustomIcon	shows a custom icon. This icon needs to be set before with the Dial_IconControl function

You can also add this switch:

-IconSize=48	display the icon at 48x48 pixels

Returned result

This function always returns 0.

Special considerations

You can hold flash dialogs on the screen longer, or get rid of them early. This is what is possible:

- To dismiss a flash dialog: click on the dialog or press the ENTER or SPACE key.
- To keep a flash dialog longer on the screen: click on the dialog and hold the mousebutton down. Or keep the ENTER or SPACE key down. When you release the mousebutton or key the dialog goes away.

IMPORTANT: Use only in a script. Do not use this function in a calculated field definition, because when FileMaker calculates the fields, this will result into a repeated dialog for EACH record.

Example usage

```
Set Field[gButtonNr, Dial_FlashDialog( "-Unused" ; 120 ; "Have a nice day")]
```

This will show a flash dialog box for 120 ticks = 2 seconds.

Example 2

We assume that in your FileMaker file the following fields are defined:

gDialogText	Global, text
gTicksPrefs	Global, text
result	Global, text

The gDialogText field contains the text you want to display, and can be filled for example with a calculation. gTicksPrefs should be filled with a time in 1/60th of a second. You can then use this as a preference. Add the following script step:

```
Set Field [ result, Dial_FlashDialog( "-NoteIcon" ; gTicksPrefs ;  gDialogText) ]
```

This shows a flash dialog with the text and a note icon.

Dial_GetButton

Syntax Dial_GetButton(switches)

Returns the number of the button clicked in the last dialog shown.

Parameters

none, please leave empty for future use.

Returned result

The number of the button clicked in the last dialog shown.

Special considerations

This function makes it easy to get the button number of a dialog that returns multiple results.

The button number is saved by the plug-in until a next dialog is displayed. Please be aware that each dialog function of the plug-in resets the number. Dialogs that have no buttons will set the button number to 0. (This is to be able to extend the functionality of these functions in the future).

When the FileMaker application stops, the saved button number is reset to 0.

See also the function Dial_GetInput.

Example usage

Set Field[gUserChoice, Dial_GetButton("")]

This will return the number of the button clicked, for example 3.

Example 2

This example will show a password dialog. We assume that in your FileMaker file the following field is defined:
result Global, text

Add the following script steps:

```
Set Field [ result, Dial_InputDialog( "-StopIcon -StartField2 -Bullet2" ; 2 ;
                    "Please enter your name and password: " ; "OK" ; "Cancel") ]
If [Dial_GetButton( "" )=1 and Dial_GetInput( "-text1" )= "sesame"]
  ... do your secret stuff here...
End If
```

Dial_GetCurrentTimestamp

Syntax Dial_GetCurrentTimestamp(switches)

The function returns the current timestamp with extra precision, in milliseconds.

Parameters

switches not used, reserved for future use. Leave blank or put "-unused"

Returned result

If successful it returns a Timestamp value, like for example: 12/10/2008 11:42:42,596492

Special considerations

Although the result includes milliseconds, the actual precision might be less, due to specifics of the computer etc. Also FileMaker may be needing variable amounts of time before returning the result. Our test shows you can expect about a precision of 1/100th of a second.

Example usage

Dial_GetCurrentTimestamp("-Unused")

This will return a timestamp of the current time. The result can be for example: 11/12/2008 11:42:42,596492. Note the extra fraction value after the comma.

Example 2

This example demonstrates how you can measure the difference in time between 2 scripts, which perform the same action, but are implemented differently. We assume that in your FileMaker file the following fields are defined:
Duration Testscript1 Global, timestamp
Duration Testscript2 Global, timestamp

Add the following script steps:

```
#Save the start time:
Set Variable [ $StartTimestamp;  Dial_GetCurrentTimestamp( "-Unused" ) ]
Perform Script [ Efficiency Testscript1 ]
Set Field [ Duration Testscript1; Dial_GetCurrentTimestamp( "-Unused" ) - $StartTimestamp ]
#Now do test 2:
#Save the start time:
Set Variable [ $StartTimestamp; Value:Dial_GetCurrentTimestamp( "-Unused" ) ]
Perform Script [ Efficiency Testscript2 ]
Set Field [ Duration Testscript2; Dial_GetCurrentTimestamp( "-Unused" ) - $StartTimestamp ]
```

Dial_GetInput

Syntax Dial_GetInput(switches)

returns the contents of one of the input fields. This makes it easy to get the result without parsing.

Parameters

switches specify which input data from the last dialog is returned.

Switches can be set to:
-Text1 this will return the text of the first input field (from the top) from the last dialog.
-Text2 this will return the text of the second input field (from the top) from the last dialog.
-Text3 this will return the text of the third input field (from the top) from the last dialog.
...
-Text15 this will return the text of the 15th input field (from the top) from the last dialog.

Returned result

The text of the specified input field, from the last dialog shown.

Special considerations

The text of the input field(s) is saved by the plug-in until a next dialog is displayed. Please be aware that each dialog function of the plug-in resets the text. Dialogs that have no input field(s) will set the text to "". (This is to be able to extend the functionality of these functions in the future).

When the FileMaker application stops, the input data is reset to "".

See also the function Dial_GetButton.

Example usage

Set Field[gUsername, Dial_GetInput("-Text1")]

This will put the data of the first input field of the last dialog shown, for example "John", in the global field gUserName.

Example 2

This example will show an an input dialog with 5 input fields. We use the Get-Input function to easily get the data out. We assume that in your FileMaker file the following fields are defined:

Name	Text
Address	Text
City	Text
Zip	Text
Country	Text
result	Global, text

Add the following script steps:

```
Set Field [ result,  Dial_InputDialog( "-NoteIcon" ; 5 ; "Please enter your personal data:" ;"OK" ; "Cancel"; ) ]
If [ Dial_GetButton( "" ) = 1 ]
  Set Field [ Name, Dial_GetInput( "-text1" ) ]
  Set Field [ Address, Dial_GetInput( "-text2" ) ]
  Set Field [ City, Dial_GetInput( "-text3" ) ]
  Set Field [ Zip, Dial_GetInput( "-text4" ) ]
  Set Field [ Country, Dial_GetInput( "-text5" ) ]
End If
```

Dial_GetInput

Example 3

This example will show a list dialog with 1 input field. We use the Get-Input function to easily get the data out. We assume that in your FileMaker file the following fields are defined:

FlavourChosen	Text
result	Global, text

Add the following script steps:

```
Set Field [ result,  Dial_ListDialog( "-NoteIcon" ; "Please select a fruit:" ; "OK"; "Cancel"; "Help"; "Stop" ;  "Apple|Pear|
Lemon") ]
Set Field [ FlavourChosen, Dial_GetInput( "-text1" ) ]
```

Dial_GetPopup

Syntax Dial_GetPopup(switches)

Returns the popup list that is currently stored in the plug-in.

Parameters

switches specifies which popuplist is returned

Switches must be one of these switches:

-Popup1	get popup list 1
-Popup2	get popup list 2
-Popup3	get popup list 3
-Popup4	get popup list 4
...	...
-Popup14	get popup list 14
-Popup15	get popup list 15

You can also add this switch:

-SecondValueList	get the second value list for that popup

Returned result

The list of popup values separated by returns.

Special considerations

If there is no popup list the result is an empty text.

Example usage

Set Field[gResult, Dial_GetPopup("-Popup1")]

This will return the popup list 1 as currently set in the plug-in. For example it may return:

Patrick
Jonathan
Brent
-
Guest

Dial_IconControl

Syntax Dial_IconControl(switches ; theIcon)

Sets the custom icon, to be used in subsequent dialogs where the switch "-customIcon" is specified.

Parameters

switches	this determines the behaviour of the function
theIcon	small JPG or PNG image to show at the left of the dialog.

Switches must be one of these:

-SetCustomIcon	sets the custom icon
-ForgetCustomIcon	forgets and frees the memory for the custom icon
-DefaultIconSize=48	This will make it the default to show all icons at 48x48, without the need to set it for each dialog function call

Returned result

If successful it returns 0.
If unsuccessful it returns an error code starting with $$ and the error code. Possible codes are:
$$-50 = parameter error.
$$-41 = not enough memory.
Other errors may be returned. (See OSErrrs for more information on error codes.)

Special considerations

You can specify an image of any size. However, as the dialogs show the icons at either 32x32 or 48x48 pixels, it is best to use one of these sizes.
You can use a custom icon with most of the functions: Dialog, ListDialog, InputDialog, BigInputDialog and FlashDialog.

If you set the custom icon again the old one is deleted. To reset the icon use the -ForgetCustomIcon switch.

Mac OS X: supports PNG (including transparency) and JPEG.
On Windows: PNG (including transparency), JPEG and GIFs are supported.

Example usage

We assume that in your FileMaker file the following fields are defined:

gCustomIcon	Global, container
gErrorCode	Global, text

gCustomIcon should contain an icon. Add the following script steps:

```
Set Field [gErrorCode, Dial_IconControl( "-SetCustomIcon" ; gCustomIcon )]
If [Left( gErrorCode ; 2 ) = "$$"]
  Beep
  Show Message [An error occurred.]
End If
```

This will set the custom icon to the contents of the container field "gCustomIcon".

Dial_InputDialog

Syntax Dial_InputDialog(switches ; nr. of fields ; prompt ; button1 ; button2; button3 ; button4)

Displays an input dialog box, in which the user can fill the input fields. Each of the up to 15 input fields can be a normal text input field, a password field, a popup menu or a checkbox.

Parameters

switches	this determines the behaviour of the dialog and which information is returned.
nr. of fields	the number of inputfields to be shown. This can be 1 to 15.
prompt	the text of the dialog
button1..4	the text of the 1st to the 4th button (from right to left)

If you specify an empty button text, no button will be displayed. Switches can be empty or you can add one or more of these switches:
-ReturnButtonText return the *text* of the button, instead of the number

To indicate which button will be selected when the user presses the ENTER key you can add one of these switches:
-DefaultButton1 -DefaultButton2 -DefaultButton3 -DefaultButton4

For each field you can also add one of these field format switches:

-Bullet1	-Bullet2	-Bullet3	-Bullet4	... -Bullet15
-Popup1	-Popup2	-Popup3	-Popup4	... -Popup15
-Checkbox1	-Checkbox2	-Checkbox3	-Checkbox4	...-Checkbox15

This indicates for a field if it is a bullet field, a popup field or a checkbox field. If you don't specify a field format switch the field will be a normal text input field.

You can also add one of these switches:

-StopIcon	shows a stop icon (indicating this is something severe which the user needs to address)
-CautionIcon	shows a caution icon (indicating this dialog warns the user)
-NoteIcon	shows a note icon (indicating this dialog gives the user information)
-CustomIcon	shows a custom icon. This icon needs to be set before with the Dial_IconControl function.

You can also add this switch:

-IconSize=48	display the icon at 48x48 pixels

You can also add this switch:
-StopOnESC

If you add this switch the the user can press the ESC key to leave the dialog. The dialog returns as button number 0.

You can also add one of these switches:
-StartField1...-StartField15 specifies the input field the cursor will be in at the start of the dialog

Returned result

The number of the button that was clicked followed by the text of the input fields. By adding switches this function can also return the text of the button.

Special considerations

You can have up to 15 input fields.
See "Dial_SetDialogTitle" if you want to change the title of the dialog.
See "Dial_SetPosition" if you want to change the position of the dialog.
See "Dial_SetInput" to set the text to be preentered into the input fields.
See "Dial_SetLabels" to set the labels to the left of the input fields.
See "Dial_SetPopup" to set the popup values to be used for popup fields.
Note: you can display popup values from a second value list. The selected value from the first popup value list is returned. You can use this for example with a list of ItemIDs and a list of ItemDescriptions. The descriptions are shown to the user, while the itemID is returned.

You can also use the keyboard shortcuts: copy (command-C), paste (command-V), cut (command-X) and select all (command-A) in the text fields.

Example usage

Set Field [result, Dial_InputDialog("-NoteIcon" ; 5 ; "Please enter your personal data: " ; "OK" ; "Cancel"; "Help" ; "Stop")]

This will show 5 input fields to fill in, with a note icon. If the user clicked the OK button a possible result can be: "1|John|Smith|Berkeley|USA|student".

Example 2

We assume that in your FileMaker file the following fields are defined:

```
result                  Global, text
gErrorCode      Global, text
```

Add the following script steps:

```
Set Field [ gErrorCode, Dial_SetLabels( "-Unused " ; "Name" ; "Password" ; "Remember password"; )]
Set Field [ gErrorCode, Dial_SetInput("-Unused" ;  Get(UserName) ; ""; "1"; )]
Set Field [ result, Dial_InputDialog( "-StopIcon -Bullet2 -Checkbox3" ; 3 ;
                "Please enter your name and password:" ; "OK" ; "Cancel"; "Help" ; "Stop" ) ]
```

This will show a user/password dialog which also has a checkbox callled "Remember password". The checkbox will be checked because the input for that field is set to 1.

Dial_ListDialog

Syntax Dial_ListDialog(switches ; prompt ; button1 ; button2; button3 ; button4 ; listitems)

Displays a list dialog box, from which the user can choose an item.

Parameters

switches this determines the behaviour of the dialog and which information is returned.
prompt the text of the dialog
button1..4 the text of the 1st to the 4th button (from the right to left)
listitems the list of items (separated by a "|" char) which the user can choose from

If you specify an empty button text, no button will be displayed.

Switches can be empty or you can add one or more of these switches:
-ReturnButtonText return the *text* of the button, instead of the number
-ReturnListText return the *text* of the listitem selected, instead of the index in the list
-AllowMultipleSelection allows the user to select multiple items from the list. Selected items are returned separated by the pipe character

You can also add one of these switches:
-DefaultButton1 -DefaultButton2 -DefaultButton3 -DefaultButton4

This indicates which button will be selected when the user presses the ENTER key.

You can also add one or more of these switches:
-Button1NeedsSelection -Button2NeedsSelection -Button3NeedsSelection -Button4NeedsSelection

If you add this switch the button is disabled until the user selects an item.

You can also add one of these switches:
-StopIcon shows a stop icon (indicating this is something severe which the user needs to address)
-CautionIcon shows a caution icon (indicating this dialog warns the user)
-NoteIcon shows a note icon (indicating this dialog gives the user information)
-CustomIcon shows a custom icon. This icon needs to be set before with the Dial_IconControl function.

You can also add this switch:
-IconSize=48 display the icon at 48x48 pixels

-Width=x makes the size of the dialog x pixels wide
-Height=y makes the size of the dialog y pixels high

You can also add this switch:
-StopOnESC
If you add this switch the the user can press the ESC key to leave the dialog. The dialog returns as button number 0.

Returned result

The number of the button that was clicked followed by the index in the list of the item that was selected. By adding switches this function can also return the text of the button and/or the item selected.

Special considerations

See Dial_SetDialogTitle if you want to change the title of the dialog.
See Dial_SetListInput if you want to preselect an item from the list.

You can type the first letter of the item you want to select.

New in v6.0: you can add the switch: -AllowMultipleSelection which allows the user to select multiple items from the list. Use the Command key on OS X or the Control + Alt key on Windows to select or deselect extra items. Or use Command-A

(on OS X) or Control-A (on Windows) to select all items.

You can also preselect multiple items with Dial_SetListInput. Separate them with a pipe character, for example "fire|water".

Example usage

Set Field [result, Dial_ListDialog("-NoteIcon" ; "Please select a fruit:" ; "OK"; "Cancel"; "Help"; "Stop" ; "Apple|Pear|Lemon")]

This will show a list of 3 items to choose from, with a note icon. If the user chooses the first item and presses the OK button "1|1" will be returned.

Example 2

We assume that in your FileMaker file the following fields are defined:

gFlavors	Global, text
result	Global, text

gFlavors should contain a list of items to choose from with each item separated by a "|". Add the following script steps:

Set Field [result, Dial_ListDialog("-CustomIcon -returnListText -Button1NeedsSelection " ; "Please select a fruit:" ; "OK"; "Cancel"; "" ; "" ; gFlavors)]

This will show the list of items to choose from. The dialog uses a custom icon. This icon needs to be set before with the Dial_IconControl function. The OK button can only be pressed after the user selected an item.

Dial_PresentImage

Syntax Dial_PresentImage(switches ; showTicks; fileList)

Displays one or more images or movies full screen or as a slideshow.

Parameters

switches determine the behaviour of the function
showticks the amount of time each image will be shown, in ticks (= 1/60th of a sec). Movies will be played to the end.
fileList one or more paths to image or movie files, separated by a return, can also be a container

switches can be one of this:
-KeepMovieControllerVisible the movie controller will be shown and not be hidden.
-PresentOn2ndScreen use the secondary display (the main being the one with the menu bar)

The fileList can be a container field with an image or movie in it. It can also be a list of one or more file specifications separated by returns.
On OS X you can use one of the following syntax:
a full path: HardDiskName:folder:folder:imagename.jpg
a FileMaker image path: imagemac:/HardDiskName/folder/folder/imagename.jpg
a FileMaker movie path: moviemac:/HardDiskName/folder/folder/moviename.mov

On Windows you can use:
a full path: C:\folder\folder\imagename.jpg
a FileMaker image path: imagewin:/C:/folder/folder/imagename.jpg
a FileMaker movie path: moviewin:/C:/folder/folder/moviename.mov

Returned result

If successful it returns 0 after the display of the images

If unsuccessful it returns an error code starting with $$ and the error code. Possible codes are:
$$-50 = parameter error
$$-41 = not enough memory
$$-4250 = QuickTime version too old

Other error codes might be returned.

Special considerations

- This function is not available in FileMaker Pro 14 and 15 (64-bit).
- This function requires QuickTime 7 to be installed.
- The maximum time an image can be shown is capped to 1 minute
- all formats supported by QuickTime can be shown, like the still image formats PICT, JPEG, GIF, TIFF, PNG, Photoshop, PDF , BMP, and movie formats like QuickTime MOV, MPG and AVI.
- Animated GIFs are played like movies.
- You can also use a container field as the fileList parameter. The container field can be an image stored in the file or stored as a reference (movies are always stored as reference).
- By default the movie controller will be hidden when showing a movie. When moving the mouse the controller will be visible for a few seconds.
- Use the ESC key to stop the display of one image or movie. Use SHIFT ESC to stop all movies.
- If the image can not be displayed an error text is displayed full screen, the slideshow will continue.
- in the script add a Allow toolbars[off] step, as toolbars will remain visible otherwise.

Example usage

Dial_PresentImage("-Unused" ; 60; "C:\sample images\myphoto.jpg")
will show the JPEG image full screen for 60 ticks = 1 second.

Dial_PresentImage

Example 2

This example will use fields to show a slideshow of images and movies. We assume that in your FileMaker file the following fields are defined:

Filelist	Text field
gErrorCode	Global, text

In the field FileList there should be a list of items to display in sequence, for example:

imagemac:/myDisk/Users/yann/coeurdevoh.jpg
myDisk:testfolder:Supertest.mov
myDisk:testfolder:images:troilogo.gif
moviemac:/myDisk/Users/spielberg/indiana.mov

Add the following script step:

Set Field [gErrorCode, Dial_PresentImage("-Unused" ; 60; Filelist)]

This will show the slideshow of the chosen images.

Dial_ProgressBar

Syntax Dial_ProgressBar(switches ; command ; maxval/incrval ; text)]

Displays a progress bar dialog box.

Parameters

switches	
command	the command to be used
maxval	the number of steps that should be used
incrval	indicates how much steps the progress bar should increase
text	the text to be shown

The first 'command' parameter indicates the commands, which may be: Show, Incr or Stop. They are to be used as follows:
1- Show command: makes the progress bar dialog box visible.
2- Incr command: increases the progress bar (multiple times, usually in a loop).
3- Stop command: the dialog is removed.

Returned result

This function returns 0 as result.

Special considerations

If you specify a zero as maxval, an indefinite progress bar is shown (barber pole).

In some cases the time to perform lengthy scripts might increase, as FileMaker updates the screen after the ProgressBar step. A freeze screen step does not always help.

IMPORTANT Use only in a script. Do not use this function in a calculated field definition, because when FileMaker calculates the fields this will result into a repeated dialog for EACH record. Also note that functions that use the clipboard, like pasting in a container field, will not work, as they try to paste in the progressbar window.

See also the Dial_SetPreference function to change the time of automatic removal of a progress dialog.

You can also suppress the title in all ProgressBar dialogs by adding this to your (startup) script:
Dial_SetPreference("-TitlesInProgressBarDialogs" ; "off")
This will set the Progress bar title to blank for all subsequent calls until FileMaker is restarted.

Example usage

Use this function to indicate to a user that a script is doing a lengthy operation and provide feedback on how long this operation will last.

For a detailed description see the section "Using the Dial_ProgressBar function" in the user guide.

Dial_SetDialogTitle

Syntax Dial_SetDialogTitle(switches ; dialogTitle)

Sets the default Dialog Title for the title bar of dialogs.

Parameters

dialogTitle the title of the dialog that you want to use for all subsequent dialogs

Returned result

An error code. Currently the plug-in always returns 0

Special considerations

The title will stay the same for all following dialogs until you change it again or restart FileMaker. If you add this function to your startup script, all dialogs with a title bar will show the title you've given.

NOTE On OS X all dialogs except the list dialog have no title bar. This function has no effect for these title bar-less dialogs.

Example usage

Dial_SetDialogTitle("-Unused" ; "Troi's Dialog Demo Solution")

This will set the title of subsequent dialogs to "Troi's Dialog Demo Solution".

Example 2

We assume that in your FileMaker file the following fields are defined:

gDialogTitle	Global, text
gResult	Global, text

Add the following script step:

```
Set Field [ gResult , Dial_SetDialogTitle( "-Unused" ; gDialogTitle ) ]
```

This script will use the contents of the global text field to set the dialog title.

Dial_SetInput

Syntax Dial_SetInput(switches ; text1 ; text2 ; text3 ; text4 ; ... ; text15)

Sets the text to be pre-entered in the input fields of the next input dialog.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
text1...text15	the text that will be pre-entered into the input fields 1 to 15.

Returned result

An error code. Currently the plug-in always returns 0.

Special considerations

The text of the input field(s) is saved by the plug-in until a dialog is displayed. Please be aware that each dialog function of the plug-in resets the text. Dialogs that have no input field(s) will set the text to "". (This is to be able to extend the functionality of these functions in the future).

When the FileMaker application stops, the input data is reset to "".

For password fields: the text is ignored (as you can't see it anyway).
For popup fields: if the text is present in the list of possible choices the first one is selected.
For checkbox fields: use 1 and 0 to check resp. uncheck a checkbox field.

Example usage

```
Set Field[ gErrorCode,
        Dial_SetInput( "-Unused" ; "Apple" ; "London" ; "16" ; "Brown" ; "Square" ; )]
```

This will set the text to be used as the pre-entered text in the next InputDialog function to the five texts.

Example 2

This example will set the pre-entered text to your user name and the value from a field. We assume that in your FileMaker file the following field is defined:

gErrorCode	Global, text
gFieldText	Global, text

Add the following script step:

```
Set Field [ result,  Dial_SetInput( "-Unused" ;  Get(UserName) ;  gFieldText ; ) ]
```

Dial_SetLabels

Syntax Dial_SetLabels (switches ; label1 ; label2 ; label3 ; ... ; label15)

Sets the label text to be displayed next to the input fields of the next input dialog.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
label1...label15	the text that will be used as labels for the input fields 1 to 15.

Returned result

An error code. Currently the plug-in always returns 0.

Special considerations

The labels of the input fields are saved by the plug-in until they are set again. So you only have to set them once if you want to use the same labels for multiple dialogs. When the FileMaker application stops, the labels are reset to "".

If you have less than 15 fields for the next dialog box you just specify the labels you need.

Note that for checkbox fields the label is put to the right of the checkbox.

Example usage

Set Field[gErrorCode, Dial_SetLabels("-Unused" " ; "Name" ; "Address" ; "City" ; "Zip" ; "Country" ;)]

This will set the labels for fields 1 to 5.

Example 2

This example will set the labels from a field, this makes it easy to translate a dialog to a different language. We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, text
gLabel1 ... gLabel15	Globals, text

Add the following script step:

Set Field [result, Dial_SetLabels("-Unused " ; gLabel1 ; gLabel2 ; gLabel3 ; gLabel4 ; gLabel5 ; gLabel6 ; ...; gLabel15)]

Dial_SetListInput

Syntax Dial_SetListInput(switches ; preselectedText)

This function allows you to preselect an item from the list in the Dial_ListDialog function.

Parameters

switches	not used, reserved for future use. Leave blank or put "-unused"
preselectedText	the text of the list dialog to be selected

Returned result

An error code. Currently the plug-in always returns 0.

Special considerations

See Dial_ListDialog for more information on the actual list dialog.

Example usage

Set Field [result, Dial_SetListInput("-Unused" ; "Lemon")]
Set Field [result, Dial_ListDialog("-NoteIcon" ; "Please select a fruit:" ; "OK" ; "Cancel"; "Help" ; "Stop" ; "Apple|Pear|
Lemon")]

This will show a list dialog with the Lemon selected when the dialog starts.

Dial_SetPopup

Syntax Dial_SetPopup(switches ; popupvalues {; secondvaluelist})

Sets the values to be used for the popup list.

Parameters

switches	specifies which popuplist is set
popupvalues	the list of text strings to be used. Separate each item with a return "¶"
secondvaluelist	(optional) second list of text, to be shown to the user, while returning the selected value from the first list.

Switches must be one of these switches:

-Popup1	set popup list 1
-Popup2	set popup list 2
-Popup3	set popup list 3
-Popup4	set popup list 4
...	...
-Popup13	set popup list 13
-Popup14	set popup list 14
-Popup15	set popup list 15

Returned result

An error code. Currently the plug-in always returns 0.

Special considerations

If you put a hyphen "-" on a separate line as a popupitem then the popup shown in the input dialog will have a separator line.

The popupvalues of this popup are saved by the plug-in until you set this popup again. So you only have to set a popup once if you want to use the same popup list for the same field in multiple dialogs. When the FileMaker application stops, the popups are reset to "".

You can limit to specifying the popups you will be using, for example popup 3 and popup 5.

Note that the number of separate popupvalues is limited to 2048. Each item can be up to 255 characters. If no items are given the popup will be disabled (grayed out).

Note: you can display popup values from a second value list. The selected value from the first popup value list is returned. You can use this for example with a list of ItemIDs and a list of ItemDescriptions. The descriptions are shown to the user, while the itemID is returned.

Example usage

Set Field[gErrorCode, Dial_SetPopup("-Popup2 " ; "Patrick¶Jonathan¶Brent¶-¶Guest")]

This will set popup list 2 to:
Patrick
Jonathan
Brent
-
Guest

All next input dialogs, which have a popup as field 2, will use this list.

Dial_SetPopup

Example 2

This example will show an input dialog with 2 popup lists. The first list is filled from a field. The second is filled from a valuelist. We assume that in your FileMaker file a value list "PopupValues" is defined and also the following fields:

gErrorCode	Global, text
gPopupData	Global, text
gInputText1	Global, text
gInputText2	Global, text
gResult	Global, text

Add the following script steps:

```
Set Field [ gErrorCode, Dial_SetPopup( "-Popup1" ; gPopupData ) ]
Set Field [ gErrorCode, Dial_SetPopup( "-Popup2" ;
                        ValueListItems( Get(FileName) ; "PopupValues" ) ) ]
Set Field [ gResult, Dial_SetInput( "-Unused" ; gInputText1 ;  gInputText2 ; ]
Set Field [ gResult, Dial_InputDialog( "-NoteIcon -Popup1 -Popup2" ; 2 ; "Please enter your personal data: " ; "OK" ;
"Cancel"; ) ]
```

Note that the popups are preselected with the values from gInputText1 and gInputText2, if that text is present in the list of possible choices.

Example 3

Say you have a database with items with an ItemID and ItemDescriptions. You want to show the ItemDescriptions to the user, while getting the ItemID back as a result. Set the popup as follows:

```
Set Variable [ $ErrorCode, Dial_SetPopup( "-Popup1" ;
"ST0001¶ST0002¶ST0003¶-¶ST0040"; // = first itemIDs, not shown in the dialog
"Pen¶Pencil¶Biro¶-¶Stapler"  // = the main text, which will be shown in the popup.
)
```

In the Dial_InputDialog the popup shows the descriptions to the user, while the selected itemID is returned.

Example 4

To clear the values from a popup list use a command like this:

```
Set Field[ gErrorCode, Dial_SetPopup( "-Popup2 "; "" ) ]
```

This will set the popup list 2 to an empty value.

Dial_SetPosition

Syntax Dial_SetPosition(switches ; left ; top)

Sets the position on the screen of dialogs to be shown.

Parameters

switches	specifies the scope of the SetPosition function
left	the left co-ordinate (in pixels) of the dialog
top	the top co-ordinate (in pixels) of the dialog

Switches can be empty or must be one of these:

-All	show all subsequent dialogs at the specified position
-Once	show only the next dialog at the specified position

If you don't specify a switch the position is set once. If you want to reset the position to the default position use this switch:

-Default	show all subsequent dialogs at the default position

Returned result

An error code. Currently the plug-in always returns 0.

Special considerations

On OS X the plug-in makes sure the dialog does not display under the menu bar.

Example usage

Set Field[gErrorCode, Dial_SetPosition("-Once" ; 100 ; 150)]

This will set the position of the next dialog box to 100 pixels from the left and 150 pixels from the top of the screen.

Example 2

This example will use fields to set the position for all next dialogs. We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, text
gLeft	Global, text
gTop	Global, text

Add the following script steps:

Set Field [gErrorCode, Dial_SetPosition("-All " ; gLeft ; gTop)]

Dial_SetPreference

Syntax Dial_SetPreference(switches ; value)

Sets preferences, like the timeout time of the progress dialog.

Parameters

switches	specifies which preference to set
value	the value to set it to.

Switches can be this:

-ProgressbarDefunctTicks	Set the time the progress bar disappears automatically in 1/60th of a second.

or it can be:

-TitlesInProgressBarDialogs	Set to "off" to suppress titles at the top of progress bar dialogs for all next calls until FileMaker is restarted. Set to "on" to show the titles again.

Returned result

If successful it returns 0.
If unsuccessful it returns an error code starting with $$ and the error code. Possible codes are:
$$-50 = parameter error.

Example usage

Set Field[gErrorCode, Dial_SetPreference("-ProgressbarDefunctTicks" ; 20* 60)]

This will change the time of automatic removal of a progressbar dialog to 20 seconds. When you forget to remove the ProgressBar window the plug-in removes it automatically after 30 seconds.
In some special cases you might want to change this, for example to be able to have a longer debug time.

Example 2

Set Variable[dontCare, Dial_SetPreference("-TitlesInProgressBarDialogs" ; "off")]

This will suppress titles at the top of progress bar dialogs for all next calls until FileMaker is restarted.

Dial_Version

Syntax Dial_Version(switches)

Use this function to see which version of the plug-in is loaded. This function is also used to register the plug-in.

Parameters

switches determine the behaviour of the function

switches can be one of this:

-GetVersionString	the version string is returned (default)
-GetVersionNumber	returns the version number of the plug-in
-ShowFlashDialog	shows the Flash Dialog of the plug-in (returns 0)
-GetPluginInstallPath	returns the path where the plug-in is installed
-GetRegistrationState	get the registration state of the plug-in: 0 = not registered ; 1 = registered
-UnregisterPlugin	sets the registration state of the plug-in to unregistered, returns 0.

If you leave the switches parameter empty the version string is returned.

Returned result

The function returns ? if this plug-in is not loaded. If the plug-in is loaded the result depends on the input parameter. It is either a:

-GetVersionString:
If you asked for the version string it will return for example "Troi Dialog Plug-in 6.5"
-GetVersionNumber:
If you asked for the version number it returns the version number of the plug-in x1000. For example version 5.8 will return number 5800.
-ShowFlashDialog:
This will show the flash dialog and then return the error code 0.
-GetRegistrationState:
returns 0 = the plug-in is not registered ; 1 = the plug-in is registered.

Special considerations

Important: always use this function to determine if the plug-in is loaded. If the plug-in is not loaded use of external functions may result in data loss, as FileMaker will return an empty field to any external function that is not loaded.

Example usage

Dial_Version("") will for example return "Troi Dialog Plug-in 4.0".

Example 2

Dial_Version("-GetVersionNumber") will return 4500 for version 4.5
Dial_Version("-GetVersionNumber") will return 4501 for version 4.5b1
Dial_Version("-GetVersionNumber") will return 2130 for version 2.1.3

So for example to use a feature introduced with version 4.5 test if the result is equal or greater than 4500.

Dial_VersionAutoUpdate

Syntax Dial_VersionAutoUpdate

Use this function to see which version of the plug-in is loaded, formatted for FileMaker Server's AutoUpdate function. Returns 8 digit number to represent an AutoUpdate version.

Parameters

none

Returned result

The function returns ? if this plug-in is not loaded. If the plug-in is loaded the result is a version number, it is returned in the format aabbccdd where every letter represents a digit of the level, so versions can be easily compared.

Special considerations

The Dial_VersionAutoUpdate function is part of an emerging standard for FileMaker plug-ins of third party vendors of plug-ins. The version number can be easily compared, when using the Autoupdate functionality of FileMaker Server.

Example usage

For example:
Dial_VersionAutoUpdate returns 05000100 for version 5.0.1
Dial_VersionAutoUpdate will return 05010203 for version 5.1.2.3

So for example to use a feature introduced with version 5.1 test if the result is equal or greater than 05010000.