Troi Activator Plug-in 3.5

USER GUIDE

March 2013

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

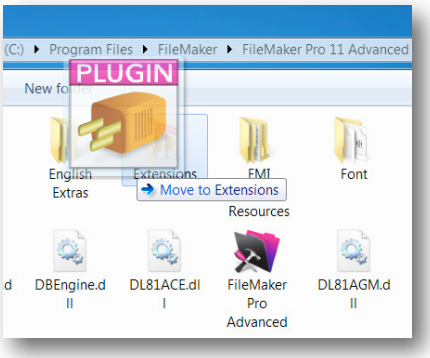
Starting with FileMaker Pro 12 a plug-in can be installed directly from a container field. Please see the **EasyInstallTroiPlugins.fmp12** example file to install plug-ins with FileMaker Pro 12.

The instructions below show FileMaker 11. You can also install the plug-in with FileMaker Pro 9 and 10.

For Mac OS X:

- Quit FileMaker Pro.
- Put the file "Troi Activator.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 “Activator Plug-in” 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 Activator Plug-in is used it will display a flash dialog box, indicating that it is loading and showing the registration status.

For Windows:



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

The instructions above show FileMaker Pro 11. You can also install the plug-in with FileMaker Pro 9 or 10.

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

You can now open the file "All Activator Examples.fp7" 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 our 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 Activator Plug-in is a very powerful tool for triggering scripts. All from within FileMaker you can:

- trigger a script on a specified date and time
- schedule events which trigger any script you want
- validate fields with a script
- notify a different user on a different computer of changes with the click of a button
- be a image upload server (Mac OS X only)

and more...

Software requirements

System requirements for Mac OS X

Mac OS X 10.5.x (Leopard), Mac OS X 10.6.x (Snow Leopard), Mac OS X 10.7 (Lion).

System requirements for Windows

Windows XP (Service Pack 3) on Intel-compatible computer, Pentium III 700MHz or faster,
Windows Vista on Intel-compatible computer, Pentium III 800MHz or faster,
Windows 7 or Windows 8 on Intel-compatible computer, 1 GHz or faster.

FileMaker requirements

FileMaker Pro 10 or FileMaker Pro Advanced 10 or higher.
FileMaker Pro 11 or FileMaker Pro Advanced 11 or higher.
FileMaker Pro 12 or FileMaker Pro Advanced 12 or higher..

NOTE Troi Activator Plug-in version 2.0 (and later) started using the native plug-in syntax introduced with FileMaker Pro 7. The functions of this plug-in have this format: FunctionName(parameter1 ; parameter2). This means that Unicode is supported and more.

Troi Activator Plug-in versions 2.0 (and later) do NOT run on versions prior to FileMaker Pro 7.0. If you need to run on versions prior to FileMaker Pro 7: see our web site for the Troi Activator Plug-in 1.5 which is using the 'classic' plug-in API, which is using the External("functionName" , "parameter") format. The 1.5 version runs on FileMaker 6, 5.x and 4.x.

It will also work with a bound runtime, created with FileMaker Advanced 9, 10, 11 or 12.

FileMaker Server and AutoUpdate

You can also use FileMaker Server 9, 10 or 11 to serve databases that use functions of the Troi Activator Plug-in. You need to have the plug-in installed at the clients that use these functions.

The AutoUpdate feature of FileMaker Server 10 and 11 can help you automate installing and updating plug-ins automatically. We created an example file and a tar formatted plug-in of Troi Activator Plug-in (only needed on Mac OS) to get you started:

Visit our AutoUpdate web page to download the example:

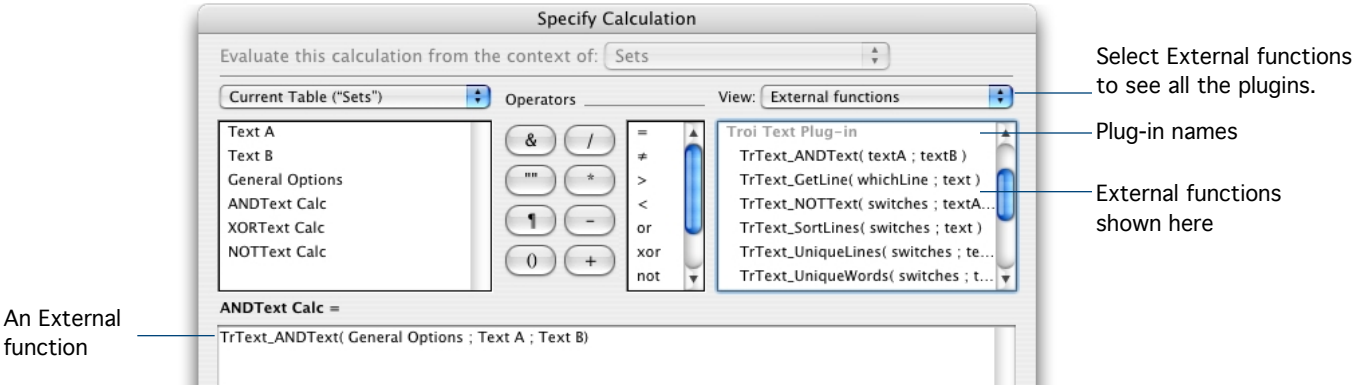
<http://www.troi.com/software/autoupdate.html>

NOTE 2 With FileMaker Pro 12 the **AutoUpdate** feature of FileMaker Server has been removed, as plug-ins can be installed directly from a container field. See the **EasyInstallTroiPlugins.fmp12** example file to install plug-ins with FileMaker Pro 12.

Getting started

Using external functions

The Activator 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 text calculation in Define Fields or in a ScriptMaker Script.

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. The function "Actr_ScheduleEvent" of this plug-in can be used in a validation calculation when you are defining fields (choose Define from the File menu). See the Validation.fp7 example.

Simple example

We start with a simple example to get you started. Say you have a database Remote.fp7, with a global text field called gMyIPAddress, in which you are going to store the IP address of the computer. Create a new ScriptMaker Script called "Get my IP address". Then add the following script step to a script:

```
Set Field[gMyIPAddress, Actr_GetIPAddress( "" )]
```

This will put the IP address of your computer into field gMyIPAddres; for example :

192.123.234.12

NOTE Function names, like "Actr_GetIPAddress " are no longer case sensitive. You can type them or get them from the External Functions list at the top right of the "Specify Calculation" dialog.

Please take a close look at the included example files, as they provide a great starting point. From there you can move on, using the functions of the plug-in as building blocks. Together they give you all the tools you need to trigger scripts from anywhere!

Triggering Scripts on other computers

First a few definitions: The receiver is the computer where a script must be triggered. The sender is the computer which sends a script trigger message to the receiver. Any computer can be a receiver and sender at the same time.

Think about security first!

Before implementing we advice you to think about the security issues, as triggering scripts is a very powerful feature. You do not want to have a "Delete All Records" script be triggered by an anonymous hacker. See below how you can limit security risks of remote triggering.

Start listening

To be able to trigger scripts on the receiver computer it must first start listening for trigger messages. If you don't want the Activator Plug-in to trigger scripts initiated by other computers you should not start listening. The plug-in will then ignore all remote trigger messages. In ScriptMaker add the following script step:

```
Set Field [ gErrorCode, Actr_StartListener(
                                "-DefaultPortNumber" ;  Get(CurrentFileName) ; "12345") ]
```

This will start the listening for messages on the default port (UDP port 54242). Only messages which have securityID 12345 are handled, other messages will be ignored. This makes listening for messages safer. No one can trigger a script on your computer, unless they know the securityID!

TIP For the securityID you can use anything you like. For example a large random number, a password or a combination of this. If you specify an empty securityID all trigger messages will be executed, even if the sender has included a non-empty securityID. See the example file "Remote.fp7" for a random number algorithm.

The receiver needs to tell other computers that it is listening for messages. A sender needs to know the IP address, the port number and the securityID (if used). A convenient way to distribute this information is via a shared FileMaker database, in which a receiver puts the needed information. See the example file Remote.fp7 where this is implemented.

NOTE This is just a suggested implementation for getting this information to the sender. It is not necessary to have a shared database for remote triggering to work. You can implement this in a different way, for example by publishing it in on the web or sending email. The information might even be always the same, so you don't need to communicate it.

TIP2 If the receiver is behind a firewall, the port on which the receiver is listening must be opened. If the receiver started listening on the default port this is UDP port 54242. See your system administrator for this.

Sending trigger messages

To send trigger scripts you only have to send the message to the receiver computer.We assume that in your FileMaker file "Remote.fp7" the following fields are defined:

Field name	Type	Possible contents
Name	Text	Peter Falk
IP Address	Text	192.1.123.24
SecurityID	Text	secretpassword
FileName	Text	Remote.fp7
ScriptName	Text	TriggerScript1
gYourID	Global, text	123
gYourText	Global, text	Please look at this record.
gErrorCode	Global, text	0

In ScriptMaker add the following script step:

```
Set Field [ gErrorCode, Actr_SendRemoteEvent("-PortNumber =51000" ;
                    IP Address ; SecurityID ; FileName ;
                    ScriptName ; gYourID ; gYourText) ]
```

This will send the message to the receiver at the specified IP address and port 51000. The receiver must be listening on this port and also the securityID must be the same. You can use the global field gYourID and gYourText to send information to the receiver, for example a record ID and a message.

Summary of functions

The Troi Activator Plug-in adds the following functions:

function name	short description
Actr_DeleteEvent	remove an event from the memory of the plug-in.
Actr_GetEventInfo	return information associated with the event, like an ID and/or text.
Actr_GetIPAddress	get the IP address of this computer.
Actr_RunScript	allows a script to be run from a calculation.
Actr_ScheduleEvent	schedule a script to be automatically triggered in the future.
Actr_SendRemoteEvent	send a trigger script message to a remote computer.
Actr_Restart	restart the computer.
Actr_Shutdown	shutdown the computer.
Actr_Sleep	put the computer into sleep; (optionally) wake it up again at a specified time.
Actr_StartListener	start listening for messages from other computers.
Actr_StopListener	stop listening for messages from other computers.
Actr_StartHTTPServer	start a HTTP Server which triggers a script when data arrives.
Actr_StopHTTPServer	stop the HTTP Server.
Actr_Version	determine which version of the plug-in is loaded; also used for registration.
Actr_VersionAutoUpdate	returns a version number for the AutoUpdate function of FileMaker Server.

Function Reference

Actr_Control

Syntax Actr_Control(switches ; password)

Controls the triggering of the plug-in. You can disable and enable local triggering of the plug-in. This allows you to change the contents of a field which is validated by triggering a script.

Parameters

switches	these determine how the function works. You can disable or enable all functions
password	the password to be used

switches can be one of the following:

-DisableLocalTriggers	disable triggering of scripts on this computer and also the scheduling of new triggers.
-EnableLocalTriggers	enable triggering of scripts on this computer and also the scheduling of new triggers.
-GetLocalTriggerStatus	get the status of the local triggering: 1 is on 0 off.

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error
$$-4217	(pwdAlreadySet) password already set (enable first)
$$-4218	(alreadyEnabled) functions are already enabled
$$-4219	(pwdWrong) given password was wrong

Other errors may be returned.

Special considerations

When you call Actr_ScheduleEvent when it is disabled an error code of $$-4220 is returned.

When you disable triggering also the scheduling of new local trigger events is disabled until restart of FileMaker. Other users can still trigger scripts on their computer and you can also send a remote event to another computer.

Example usage

Set Field[result, Actr_Control("-DisableLocalTriggers " ; "secret password")]

This will disable the trigger functions of the plug-in on this computer.

Example 2

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

gErrorCode	Global, number
gPassword	Global, text

gPassword should contain a password, for example "rapunsel". In ScriptMaker add the following to the steps to a script:

```
Set Field[gErrorCode, Actr_Control("-DisableLocalTriggers";  gPassword)]
#... do for example a Replace function  on a script validated function...
Replace Contents[ValidateField ,  newValue]
Set Field[gErrorCode, Actr_Control("-EnableLocalTriggers" ; gPassword)]
```

This will first disable the plug-ins triggers. Then you can change the values. The last step reactivates triggering.

Actr_DeleteEvent

Syntax Actr_DeleteEvent(switches ; eventID)

This function will remove an event from the memory of the plug-in.

Parameters

switches	determines which information is to be returned
eventID	(optional) the event to delete

switches can be one of this:

-DeleteLastTriggered	delete the event that was last triggered
-DeleteAllTriggeredEvents	delete all triggered events
-DeleteAllEvents	delete all events
-DeleteByID	delete the event specified in the next parameter

Returned result

If successful it returns 0. If unsuccessful it returns an error code starting with $$ and the error code. Possible codes are:
0 = no error
$$-50 = parameter error, check if your parameters are correct.
$$-41 = not enough memory
Other errors may be returned.

Special considerations

You can also delete events when you are retrieving data from the event. See the function: Actr_GetEventInfo. for this.

Example usage

Set Field[result, Actr_DeleteEvent("-DeleteAllTriggeredEvents" ;)]

This command will remove all triggered events.

Example 2

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

gErrorCode	Global, text
gEventID	Global, number

gEventID contains an eventID. In ScriptMaker add the following script step to your Trigger script:

Set Field[gErrorCode, Actr_DeleteEvent("-DeleteByID" ; gEventID)]

This will remove the event with the same EventID as in field gEventID from the memory of the plug-in.

Actr_GetEventInfo

Syntax Actr_GetEventInfo(switches ; eventID)

This function will return information that was previously associated with the event, like yourID and yourText.

Parameters

switches	determines which information is to be returned
eventID	(optional) the event to retrieve info from

switches must contain only one of this:

-GetYourID	retrieves the yourID data
-GetYourText	retrieves the yourText data
-GetActivatorEventID	retrieves the internal EventID
-GetAllEventsList	retrieves a tab separated list of all the events and the data of each event
-GetAllEventIDs	retrieves a list of all the eventIDs separated by returns
-GetImage	get the uploaded image (for the Upload Server functionality)
-GetLocationData	get the (GPS) location (for the Upload Server functionality)

and also switches must contain one of this:

-LastTriggered	retrieves the information from the event that was triggered last
-ByEventID	retrieves the information from the event with the eventID in the next parameter
-FirstSilentEvent	retrieves the information from the first silent event

optionally switches can also contain this:

-DeleteThisEvent	(optional) delete the event after the data is returned

Returned result

The requested information.

Example usage

Set Field[result, Actr_GetEventInfo("-LastTriggered -GetYourID -DeleteThisEvent" ;)]

This command will return the YourID data for the last event triggered. The event data will be deleted.

Example 2

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

gEventID	Global, number
gEventText	Global, text

gEventID contains the eventID. In ScriptMaker add the following script step to your Trigger script:

Set Field[gEventText, Actr_GetEventInfo("-ByEventID -GetYourID" ; gEventID)]

This command will fill the field gEventText with the text that was supplied earlier, when the event was created.

Actr_GetIPAddress

Syntax Actr_GetIPAddress(switches)

Gets the IP address(ses) of this computer.

Parameters

switches You can leave this blank or put one of this:

-InterfaceIndex = x	get the IP address of interface x, the first interfaces starts at 1
-Defaultinterface	get the default IP address of this computer
-GetIPv6	(Mac OS X only) return the IPv6 address

Returned result

the IP address of this computer.

Special considerations

To publish your address you can put the returned IP address in a shared database, together with other clarifying data, like your name. This makes it easy to send a remote event to a particular person: Look up the IP addres that is in the same record as the name. See the Remote.fp7example file, where this is worked out.

NOTE: On computers with multiple IP addresses GetIPaddress returns the first IP address if you leave the swicth empty. On Mac OS you can (starting with version 1.3.2) also ask for other than the first IP address.

Example usage

Set Field[result, Actr_GetIPAddress("")]

This command will return the IP address for this computer, for example "192.168.1.24".

Example 2

Set Field[result, Actr_GetIPAddress("-InterfaceIndex = 1")]
Set Field[result, Actr_GetIPAddress("-InterfaceIndex = 2")]

This command will return the first and the second IP address for this computer, for example "192.168.1.24" and "198.123.32.1".

Actr_Restart

Syntax Actr_Restart(switches)

Restart the computer

Parameters

switches determines the behaviour of this command

switches can be left empty or set to this:
-Force (currently Windows only) force other running applications to close

Returned result

The returned result is an error code. If successful it returns 0. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:
0 = no error
$$-50 = parameter error, check if your parameters are correct

Other errors may be returned.

Special considerations

If you don't specify the -Force switch the computer may not shutdown and restart, because of other programs not closing.
There is a 20 second delay before the start of the restart.
The -Force switch currently does not have any effect on Mac OS X.

Example usage

Set Field[result, Actr_Restart("")]

This will restart the computer.

Set Field[result, Actr_Restart(" -force ")]

This will restart the computer, forcing all programs to close.

Actr_RunScript

Syntax Actr_RunScript(switches ; fileName; scriptName; scriptParam)

This function will allow a script to be run from a calculation.

Parameters

switches	not used, reserved for future use. Leave blank or put "-unused"
fileName	the name of the file that contains the script
scriptName	the name of the script
scriptParam	(optional) a free to use text that will be the incoming script parameter

Returned result

If successful an eventID is returned:

eventID this is an internal eventID that the plug-in assigns, for example 4242

If unsuccessful it returns an error code starting with $$ and the error code. Possible codes are:
$$-50 = parameter error, check if your parameters are correct.
$$-41 = not enough memory
Other errors may be returned.

Special considerations

NOTE: the plug-in does not store the information of this function, so you don't need to delete anything after the script has triggered.

See the Tooltips.fp7 example for hints how to implement this.
See Actr_ScheduleEvent if you want to schedule a script trigger in the future.

Example usage

Set Field[result, Actr_RunScript("-unused" ; "Tooltips.fp7" ; "RunScript1" ; "hello")]

This command triggers the script "RunScript1" in file "Tooltips.fp7". Note that this example is somewhat simplified, normally you should not use hardcoded dates like in the above example. See also example 2, for a more robust example.

Example 2

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

gEventID	Global, number

In ScriptMaker add the following script step:

Set Field[gEventID, Actr_RunScript("-unused" ; Get(FileName) ; "RunScript1" ; "Check record please.")]

This command will directly trigger the script "RunScript1" in the current file.

Actr_ScheduleEvent

Syntax Actr_ScheduleEvent(switches ; datetimestamp; fileName; scriptName; yourID ; usertext)

This function will schedule a script to be automatically triggered in the future.

Parameters

switches	set to "-AddSingleEvent" (in the future new switches may be added)
datetimestamp	the timestamp indicating the date and time the script must be triggered
fileName	the name of the file that contains the script
scriptName	the name of the script
yourID	(optional) a free to use ID that you can supply to identify the event
usertext	(optional) a free to use text that you can supply for your own needs

Returned result

If successful an eventID is returned:

eventID this is an internal eventID that the plug-in assigns, for example 4242

If unsuccessful it returns an error code starting with $$ and the error code. Possible codes are:
$$-50 = parameter error, check if your parameters are correct.
$$-41 = not enough memory
$$-4216 = OLE error (windows)
$$-4223 = maximum number of events scheduled (1000)
$$-4226 = you can't schedule an event in the past

Other errors may be returned.

Special considerations

The date parameter should be in the same format as a FileMaker date field.
The time parameter should be in the same format as a FileMaker time field.

IMPORTANT: Don't forget to delete triggered events in the triggered scripts, as this might fill up the memory. See Actr_DeleteEvent.

NOTE: Starting with version 1.3 this function now works for all version of FileMaker Pro and later and bound runtimes from FileMaker Developer on all platforms.

Example usage

```
Set Field[result, Actr_ScheduleEvent(
        "-AddSingleEvent" ; Timestamp("12/16/2005" ; "12:22:45 PM") ;  Events.fp7" ; "TriggerScript1" ; "12345" ;
"hi!")]
```

This command triggered the script "TriggerScript1" in file "Events.fp7" on day December 16th, 2005, at 12:22:45 PM. Note that this example is somewhat simplified, normally you should not use hardcoded dates like in the above example, as this will only work on a system with US dates. See also example 2, for a more robust example.

Example 2

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

gTriggerDate	Global, date
gTriggerTime	Global, time
gEventID	Global, number

gTriggerDate and gTriggerTime can be filled with the date and time the script should be triggered. In ScriptMaker add the following script step:

```
Set Field[gEventID, Actr_ScheduleEvent( "-AddSingleEvent" ; Timestamp(gTriggerDate ; gTriggerTime) ; Get
(CurrentFileName) ; "TriggerScript" ;  Get(CurrentRecordID) ; "Check record please.")]
```

Actr_ScheduleEvent

This command will trigger the script "TriggerScript" in the current file. It will be triggered on the day and time that are in the global fields gTriggerDate and gTime.

Actr_SendRemoteEvent

Syntax Actr_SendRemoteEvent(switches ; IP_address ; securityID ; fileName ; scriptName ; yourID ; usertext)

Send a message to the computer with this IP address. This message will trigger the specified script.

Parameters

switches	see below
IP_address	the IP address of the computer that must be triggered
securityID	(optional) an ID that must be equal to the securityID of the receiving computer
fileName	the name of the file that contains the script
scriptName	the name of the script
yourID	a free to use ID that you can supply to identify the event
usertext	a free to use text that you can supply for your own needs

switches can be one of this:

-DefaultPortnumber	use the default port number of the Activator (UDP port 54242)
-PortNumber =<portnumber>	use the specified portnumber

Additional you can add:

-DontGoToForeground	the script triggers whithout the plug-in first bringing FileMaker (or Runtime) to the foreground.
-SilentEvent	do not trigger a script, just send the data

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. There was an error with the parameters.
$$-4228 = portnumber out of range: use 0 to 65535.

Other errors may be returned. See our OSErrrs database for explanations.

Special considerations

You can omit both portnumber switches. In this case the default portnumber 54242 is used.

The securityID makes listening for messages safe. No one can trigger a script on a remote computer, unless they know the securityID. If the sender sends a different securityID no triggering occurs! See "Actr-StartListener" for more information on SecurityID.

The maximum length of a remote message is 6000 characters. To accomodate the name of the script etc, this means that you should not send usertext is that is longer than 5000 characters.

When you add the -SilentEvent switch, the receiver application won't be notified, so you need to make sure the event data is retrieved, with the -FirstSilentEvent switch in the Actr-GetEventInfo function. Also dont forget to delete this event.

The -DontGoToForeground switch adds the possibility to do background processing, for example with a FileMaker Runtime application. See the example in the Background Processing folder.

NOTE Starting with version 1.3 this function now works for all version of FileMaker Pro 4 and later and bound runtimes from FileMaker Developer 4 , 5.x and 6 on all platforms.

Example usage

```
Set Field [ gErrorCode, Actr_SendRemoteEvent(
      "-DefaultPortnumber ; 192.168.1.1 ; secretword ; Filename.fp7 ; Triggerscript1 ; 1234567 ; Hello there!") ]
```

Actr_SendRemoteEvent

Example 2

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

gIPAddress	Global, text
gSecurityID	Global, text
gYourID	Global, text
gYourText	Global, text
gErrorCode	Global, text

gIPAddress and gSecurityID should contain the IP address resp. the SecurityID of the computer you want to trigger. gYourID and gYourText can be filled with the message you want to send to the other computer. In ScriptMaker add the following script step:

Set Field [gErrorCode, Actr_SendRemoteEvent("-PortNumber =51000 ; gIPAddress ; gSecurityID " ; "Remote.fp7" ; "Triggerscript1" ; gYourID ; gYourText)]

This will trigger script "Triggerscript1" in database "Remote.fp7" on the remote computer.

Actr_Shutdown

Syntax Actr_Shutdown(switches)

Shutdown the computer

Parameters

switches determines the behaviour of this command

switches can be left empty or set to this:
-Force (currently Windows only) force other running applications to close

Returned result

The returned result is an error code. If successful it returns 0. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:
0 = no error
$$-50 = parameter error, check if your parameters are correct

Other errors may be returned.

Special considerations

If you don't specify the -Force switch the computer may not shutdown, because of other programs not closing.
There is a 20 second delay before the shutdown starts.
The -Force switch currently does not have any effect on Mac OS X.

Example usage

Set Field[result, Actr_Shutdown("")]

This will shutdown the computer.

Set Field[result, Actr_Shutdown(" -force")]

This will shutdown the computer, forcing all programs to close.

Actr_Sleep

Syntax Actr_Sleep(switches ; datetimestamp)

Put the computer into sleep mode (if present). You can optionally specify when to wake up again.

Parameters

switches	determines the behaviour of this command
datetimestamp	(optional) the timestamp indicating the date and time the computer must wake up

switches can be one of this:

-SleepNow	sleep now, the computer will not wake automatically
-TillDateTime	sleep until the specified date/time

Returned result

The returned result is an error code. If successful it returns 0. An error always starts with 2 dollars, followed by the error code. You should always check for errors.

Returned error codes can be:
0 = no error
$$-50 = parameter error, check if your parameters are correct
$$-4126 = the date specified should be in the future (kErrDateInPast)
$$-4131 = this computer does not have suitable hardware to do this

Other errors may be returned.

Special considerations

- At the moment the time to wake is does NOT work, you can only put the computer to sleep.
- This function works on computers that support sleep.
- The function returns before the computer has gone into sleep. Use a Pause/Resume step with 20 seconds pause, to be sure the next step is executed after the sleep.
- Waking up starts at the specified time. It might take a few seconds for the computer to be completely awake. If it is critical that the computer is awake at a specified time, subtract 30 seconds from the wake time.

Example usage

Set Field[result, Actr_Sleep("-sleepnow")]

Put the computer to sleep. It will only wake up by the user waking it.

You can also use:

Set Field[result, Actr_Sleep("-tilldatetime"; Timestamp("12/16/2015" ; "12:22:45 PM")]

This last command put the computer into sleep and woke it up on December 16th, 2005, at 12:22:45 PM. Note that this example is somewhat simplified, normally you should not use hardcoded dates like in the above example, as this will only work on a system with US dates. See also example 2, for a more robust example.

Example 2

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

gWakeTimeStamp	Global, timestamp
result	Global, text

Actr_Sleep

gWakeTimeStamp can be filled with the date and time the computer must wake up. In ScriptMaker add the following script step:

```
Set Field[result,  Actr_Sleep("-TillDateTime" ;  & gWakeTimeStamp)]
```

This command will put the computer to sleep and wake the computer on the day and time that are in the global fields "gWakeTimeStamp".

Actr_StartHTTPServer

Syntax Actr_StartHTTPServer(switches ; fileName ; scriptName ; {password})

Starts the built-in HTTP Server. This web server will trigger a script when data arrives.

Parameters

switches	determines the behaviour of this command
filename	the name of the file which contains the script to trigger when a HTTP request arrives
scriptname	specifies the name of the script to trigger when a HTTP request arrives
password	(optional) a password which the sender needs to add to the HTTP request

switches must contain only one of this:

-Defaultportnumber	use the default port number of the Activator (TCP port 54242)
-Portnumber =<portnumber>	use the specified portnumber

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	
$$-50	paramErr	there was an error with a parameter

Other errors may be returned.

Special considerations

This functionality is currently only available on Mac OS X.

Only HTTP requests which are posted to the right web page will trigger the script. Other requests (for example for the index.html page) are handled by the plug-in. You can now also send text only to the plug-in.

The plug-in will make the HTTP server known via Bonjour as "Troi Activator Upload Server", allowing for easy discovery. See the UploadData.fp7 example file for more detailed information.

Example usage

Set Field[result, Actr_StartHTTPServer("-Unused" ; "Upload.fp7" ; "HTTP_TriggerScript" ; "secret")]

This command start the HTTP Server. When (image) data comes in it will trigger the script "HTTP_TriggerScript" in file "Upload.fp7". Note that this example is somewhat simplified, normally you should not use hardcoded files like in the above example.

Example 2

In ScriptMaker add the following script steps:

```
Set Variable[$password, "secret")]
Set Variable[result, Actr_StartHTTPServer( "-Portnumber=12345" ; Get(FileName) ; "HTTP_TriggerScript" ;
$password)]
```

This command will start the HTTP Server, with the HTTP Server listening on port 12345. Users can now send images or text data from other computers or for example a iPhone or iPad.

Actr_StartListener

Syntax Actr_StartListener(switches ; currentFileName ; securityID)

Start listening for messages from other computers.

Parameters

switches	determines the behaviour of this command
currentFileName	the name of the file that contains the script that must be triggered later
securityID	(optional) a security ID which the sender needs to add to sent messages

switches must contain only one of this:

-Defaultportnumber	use the default port number of the Activator (UDP port 54242)
-Portnumber =<portnumber>	use the specified portnumber

Returned result

If successful it returns 0.
If unsuccessful it returns an error code starting with $$ and the error code. Possible codes are:
0 = no error
$$-50 = parameter error. There was an error with the parameters.
$$-4215 = invalid FileMaker Version. On windows you need FileMaker 5 or later.
$$-4227 = already listening.
$$-4228 = portnumber out of range: use 0 to 65535.

Special considerations

Which port you should choose is dependent on your network situation: You should use a port number that is not in use. Try to use the default port number. If this port is occupied choose a number in the private range: 49152 - 65535.

Below you find how ports are currently assigned:

0 - 1023:	Well Known Ports, used by standard protocols. Don't use for Activator.
1024 - 4915:	Registered Ports. Not recommended for Activator.
49152 - 65535:	Dynamic and/or Private Ports.

The securityID makes listening for messages safe. No one can trigger a script on your computer, unless they know the securityID. If the sender sends a different securityID no triggering occurs!
For the securityID you can for example use (random) numbers or a password, and you can distribute it in a field of a shared database. See the example file "Remote.fp7". If you specify an empty securityID all messages will be triggered, even if the sender has included a non-empty securityID.

NOTE Starting with version 1.3 this function now works for all version of FileMaker Pro and bound runtimes from FileMaker Developer on all platforms.

Example usage

Set Field [gErrorCode, Actr_StartListener("-defaultportnumber" ; Get(FileName) ; "12345")]

This will start the listening for messages. Only messages which have securityID 12345 are handled, others will be ignored.

Example 2

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

gSecurityID	Global, text
gErrorCode	Global, text

gSecurityID should contain the security ID, for example "secretpassword". In ScriptMaker add the following script step:
Set Field [gErrorCode, Actr_StartListener("-portnumber=50505" ; Get(FileName) ; gSecurityID)]

This will start the listening for messages on port 50505. Only messages which are sent to this port and have securityID that's equal to the value in field gSecurityID are handled, others are ignored.

Actr_StopHTTPServer

Syntax Actr_StopHTTPServer(switches)

Stop the HTTP Server.

Parameters

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

Returned result

If successful it returns 0.
If unsuccessful it returns an error code starting with $$ and the error code.

Special considerations

Currently only available on Mac OS X.

See the UploadData.fp7 example file for more information.

Example usage

Set Field [gErrorCode, Actr_StopHTTPServer("-Unused")]

Actr_StopListener

Syntax Actr_StopListener(switches)

Stops listening for messages from other computers.

Parameters

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

Returned result

If successful it returns 0.
If unsuccessful it returns an error code starting with $$ and the error code.

Special considerations

At the moment this function only returns 0.

Example usage

Set Field [gErrorCode, Actr_StopListener("")]

Actr_Version

Syntax Actr_Version(switches)

Use this function to see which version of the plug-in is loaded.
Note: 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 = unregistered ; 1 = registered
-UnregisterPlugin sets the registration state of the plug-in to unregistered

If you leave the 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:

VersionString:
If you asked for the version string it will return for example "Troi Activator Plug-in 3.5"

VersionNumber:
If you asked for the version number it returns the version number of the plug-in x1000. For example version 3.5 will return number 3.500.

ShowFlashDialog result:
This will show the flash dialog and then return the error code 0.

GetRegistrationState result:
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.
See the Version.fp7 example file.

Example usage

Actr_Version("") will for example return "Activator Plug-in 3.1"

Example 2

Actr_Version("-GetVersionNumber") will return 3500 for version 3.5.
Actr_Version("-GetVersionNumber") will return 4001 for version 4.0b1
Actr_Version("-GetVersionNumber") will return 3120 for version 3.1.2

So for example to use a feature introduced with version 3.5 test if the result is equal or greater than 3500.

Actr_VersionAutoUpdate

Syntax Actr_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 Actr_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:
Actr_VersionAutoUpdate returns 02060100 for version 2.6.1
Actr_VersionAutoUpdate will return 03050204 for a (possible future) version 3.5.2.4

So for example to use a feature introduced with version 3.1 test if the result is equal or greater than 03010000.