Troi URL Plug-in 4.5

for FileMaker Pro 15

USER GUIDE

May 2016

troi automatisering

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

Table of Contents (continued)

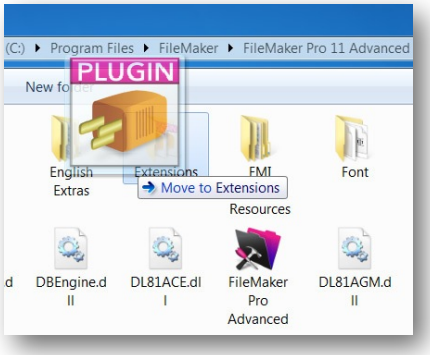
Installing plug-ins

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 and 15.
The instructions below are for FileMaker Pro 11.

For Mac OS X:

- Quit FileMaker Pro.
- Put the file "Troi_URL.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_URL.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 Troi URL Plug-in is used it will display a dialog box, indicating that it has loaded and showing the registration status.

For Windows:



- Quit FileMaker Pro.
- Put the file "Troi_URL.fmx" from the directory "Windows Plug-in" into the "Extensions" 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_URL.fmx'. Would you like to replace the existing file with this one?’. Press the Yes button.
- Start FileMaker Pro. The first time Troi URL Plug-in is used it will display a dialog box, indicating that it has loaded 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 **Edit** menu, and then choose **Plug-ins**.

You can now open the file "All URL Examples.fmp12" to see how to use the plug-in's functions. There is also a function overview in the download.

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. You may also 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?

Troi URL Plug-in lets you post and retrieve information from the Internet. Troi URL Plug-in can help you fill in web forms on the Internet, all from FileMaker Pro. It also retrieves data from any HTTP URL.
You can:

- use the POST command to fill in a web form and retrieve the result directly in FileMaker
- use the GET command to retrieve data
- use a secure connection (HTTPS) using SSL
- use cookies, custom headers and get data via a proxy server
- easily authorize an account for twitter and post tweets from FileMaker.

This plug-in makes it possible to:

- fill in forms on web sites or your intranet
- get online data like stock quotes, weather reports, etc.
- automatically add information to listings and online databases
- get images from web sites into FileMaker containers

Software requirements

System requirements for Mac OS X

Mac OS X 10.6.8 (Snow Leopard), OS X 10.7 (Lion), OS X 10.8 (Mountain Lion), OS X 10.9 (Mavericks) OS X 10.10 (Yosemite), OS X 10.11 (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 Pro requirements

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

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

Troi URL Plug-in 4.5 will also work with a bound runtime, created with FileMaker Advanced 12, 13, 14 or 15.

FileMaker Server requirements

FileMaker Server 12 or FileMaker Server Advanced 12 or higher.
FileMaker Server 13 or FileMaker Server Advanced 13 or higher.
FileMaker Server 14 or FileMaker Server Advanced 14 or higher.
FileMaker Server 15 or FileMaker Server Advanced 15 or higher.

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

Troi URL Plug-in can also be used by FileMaker Server as a server-side plug-in or as a plug-in used by the web publishing engine. To use Troi Plug-ins as a server-side or web-side plug-in you need to purchase a special Server/Web license. More information can be found in the download or here:

http://www.troi.com/support/filemaker-server-side-plug-ins.html

FileMaker Server and AutoUpdate

With FileMaker Pro 12 the AutoUpdate feature is no longer needed, as plug-ins can can be installed directly from a container field (See the **EasyInstallTroiPlugins.fmp12** example file to install plug-ins with FileMaker Pro 12 to 15.)

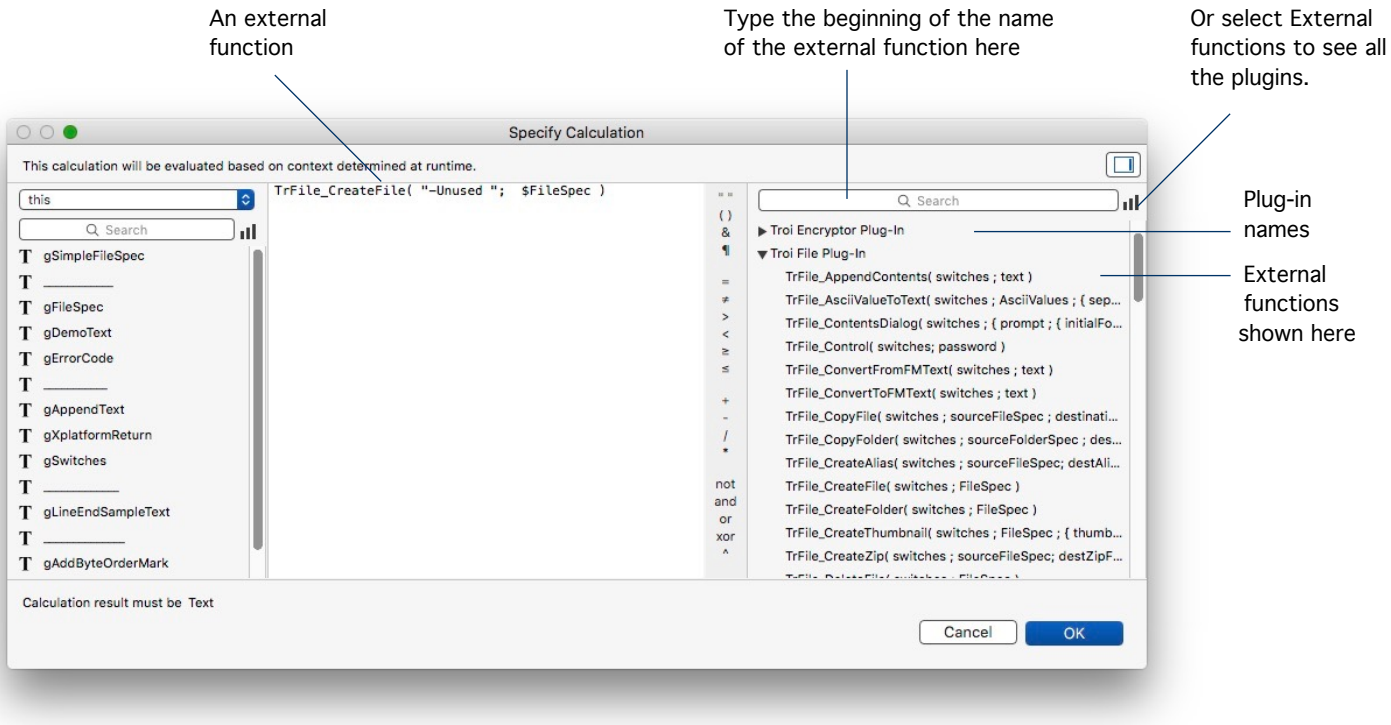
If you still use FileMaker Server 11 you can use the AutoUpdate feature of FileMaker Server 11 to help you automate installing and updating plug-ins automatically. We created an example file and a tar formatted plug-in of Troi URL Plug-in (only needed on Mac OS X) to get you started. Visit our AutoUpdate web page to download the example:

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

Getting started

Using external functions

Troi URL 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 Set Field 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. Only the TURL_ToHTTP function has no side effects and can be used in a define field calculation.

Simple example

This example shows how to begin using the Troi URL Plug-in. Say you have a database myGetTest.fmp12, with a global text field called gResult. Create a script "Simple Example". Add the following script step to this script:

```
Set Field[gResult, TURL_Get("-unused" ; "http://www.example.com/") ]
```

This will get the web page and put it in the field gResult.

The result will be similar to this:

```
<HTML><HEAD><TITLE>Example Web Page</TITLE></HEAD>
<BODY><p>You have reached this web page by typing www.example.com ...These domain
names are reserved for use in documentation and are not available for registration.</
p> </BODY>/HTML>
```

Function names, like `TURL_Get`, are not 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 interact with the Internet in powerful ways directly from FileMaker Pro.

You can use globals or variables

It is possible to use variables in calculations. Our example files in the download now both use global fields and variables to pass parameters and store the results of a plug-in function.

As this release of Troi URL is intended for FileMaker Pro 12 and higher, we will use variables wherever possible. Note that not all examples are using variables yet.

All plug-in functions work with variables just fine. For example if you have these script steps:

```
Set Field [this::gURL,  "your url here"  ]
Set Field [this::gResult,  TURL_Get(  "-NoDialog" ; this::gURL )  ]
```

With variables you can alternative use:

```
Set Variable [$URL,  "your path here"  ]
Set Variable [$Result,  TURL_Get(  "-NoDialog" ; this::gURL ; $URL )  ]
```

The main advantage of variables is that you don't need to define global fields that clutter your database definitions. The variables can stay local to the script.

Summary of functions

Troi URL Plug-in adds the following functions to FileMaker Pro:

function name	short description
TURL_Version	Use this function to see which version of the plug-in is loaded. This function is also used to register the plug-in.
TURL_Get	Gets the raw data of the specified URL. This can be for example the HTML of a web page.
TURL_Post	Sends data to the server, using the POST method, and returns the result as raw data.
TURL_Put	Sends data to the server or transfers a file to the server, using the PUT method.
TURL_Delete	Sends request to delete an item at a specified place on the web server, using the DELETE method.
TURL_GetLastHTTPStatusCode	Returns the HTTP response status code from the last executed HTTP request.
TURL_SetUserName	Sets the user name to be used for password protected URLs.
TURL_SetPassword	Sets the password to be used for password protected URLs.
TURL_SetCookies	Sets the cookies text to be used to send to a web server.
TURL_SetCustomHeader	Sets a custom header to be used.
TURL_SetUserAgent	Sets a custom text for the name of the user agent the plug-in sends to a server.
TURL_SetProgressText	Sets a custom text to be used for the progress dialog.
TURL_SetProxy	Sets the proxy server to be used.
TURL_SetAuthCredentials	Sets the Consumer Key and Consumer Secret for an OAuth authorization.
TURL_GetAuthorizationURL	First step to authorize for a web service: gets an authorization URL.
TURL_SendAuthorizationPIN	Second step to authorize for a web service: sends the PIN to the web service to complete the authorization.
TURL_SendAuthorizedRequest	Sends a request (like posting a tweet), using credentials obtained during authorization.
TURL_IsSecure	Indicates if the last request was secure.
TURL_ToHTTP	Encodes a text in HTTP format.
TURL_ToURLEncoded	URL encodes text (also known as percent-encoding).
TURL_HMACSHA1	Calculates a keyed-hash message authentication code (HMAC-SHA1) signature.

How to make sense of a web form

This section describes how to determine what you must send to make a web form work.

Introduction

On the Internet there are a lot of web forms. A web form requires one or more fields to be filled before you can use a submit button to send the data to the server. The web server then sends a response. Web forms usually work with the POST method of the HTTP protocol.

It is not that easy to find out the right way to send form data to the server. You need some knowledge of HTML. To find out what to send to a server use the following steps:

1 Save the HTML source of the page.
2 Find the <FORM> part in the source
3 Remove all formatting, leaving only the fields of the form
4 Build the form data to send

Step 1: Save the HTML source of the page

In your web browser load the page with the web form on it. Then save the HTML source. On some browsers this can be done with the "Save as" command under the File menu. Choose as format "HTML source" or "Web Page, HTML only" or similar.

Step 2: Find the <FORM> part in the source

Open the file in a Text editor, like BBEdit on Mac or Notepad on Windows. Or use a HTML editor if you have one. The page will look something like the simplified page in figure 1.

```
<HTML>
<HEAD>
   <TITLE>Test Form</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<H2>An example form</H2>

<P><FORM ACTION="/cgi-sys/formmail.pl" METHOD=POST>
   <P><INPUT TYPE=text NAME=field1 VALUE="" SIZE=30><FONT   COLOR="#FF0000"><I> </
I></FONT><INPUT TYPE=text NAME=field2 VALUE=""       SIZE=30><INPUT TYPE=hidden
NAME=secret VALUE=A12474><INPUT TYPE=submit    NAME=Submit VALUE="Submit for Processing">
</FORM></P>

<P><I>&copy; 2011 Troi Automatisering, Have Fun!</I></P>
<P></P>
</BODY>
</HTML>
```

Figure 1: Example page with form (simplified)

Find the form, by locating the <FORM> tag. The end of the form is indicated with the </FORM> tag. In our example page this is easy, but most web pages are much more complicated.

NOTE there may be more forms on that web page. Make sure you find the correct form.

Remove everything outside the <FORM> </FORM> tags. In our example this will results in figure 2:

```
<FORM ACTION="/cgi-sys/formmail.pl" METHOD=POST>
   <P><INPUT TYPE=text NAME=field1 VALUE="" SIZE=30><FONT
COLOR="#FF0000"><I> </I></FONT><INPUT TYPE=text NAME=field2 VALUE=""
SIZE=30><INPUT TYPE=hidden NAME=secret VALUE=A12474><INPUT TYPE=submit
NAME=Submit VALUE="Submit for Processing">
</FORM>
```

Figure 2: Only the form

Step 3: Remove all formatting, leaving only the fields of the form

Remove all formatting tags, and irrelevant text etc. You should leave all the <INPUT ...> tags. With some extra returns this gives:

```
<FORM ACTION="/cgi-sys/formmail.pl" METHOD=POST>
<INPUT TYPE=text NAME=field1 VALUE="" SIZE=30>
<INPUT TYPE=text NAME=field2 VALUE="" SIZE=30>
<INPUT TYPE=hidden NAME=secret VALUE=A12474>
<INPUT TYPE=submit NAME=Submit VALUE="Submit for Processing">
```

Figure 3: The form without formatting

Step 4: Build the form data to send

In the ACTION attribute of the FORM tag you can find where the form data should be sent, in this case: "/cgi-sys/formmail.pl". Assuming that this page is run on the web site www.example.com the total URL would be: "http://www.example.com/cgi-sys/formmail.pl" Note also the METHOD=POST attribute. This can also be METHOD=GET.

In the form itself you now see there are 3 INPUT fields and a submit field.

field1 with an empty initial value
field2 with an empty initial value
secret with value A12474
Submit with value Submit for Processing

In the POST method you need to concatenate all fields separated with an ampersand "&".

From all this we can infer the data to send. Assuming we want to send data1 as the value for field1 and data2 as the value for field2, then the form data to be sent should be the text:

field1=data1&field2=data2&secret=A12474&Submit=Submit%20for%20Processing

NOTE Spaces are not allowed, so you need to encode these as %20. Also other higher ASCII characters need to be encoded. You can use the TURL_ToHTTP function of the plug-in to do this.

To finally bring it all together you create a ScriptMaker Script. In this ScriptMaker Script you create this script step:

```
Set Field [result ,
   TURL_Post("-unused" ;
   "http://www.example.com/cgi-sys/formmail.pl" ;
   "field1=data1&field2=data2&secret=A12474&Submit=Submit%20for%20Processing")]
```

This will POST the form data to the web server and return the result.

Using fields as data

You can also get the data to send to a server from fields in the database. We assume that you want to send to the same web form as above. In your FileMaker file the following fields should be defined:

result	Text
field1	Text
field2	Text
formURL	Text
gTimeOutTime	Global, number

The fields field1 and field2 should be filled with the data to send to the server. The field formURL will be set to the URL that handles the form. In ScriptMaker add the following script steps:

```
Set Field [formURL , "http://www.example.com/cgi-sys/formmail.pl"]
Set Field [result  ,
   TURL_Post("-unused" ; formURL ;
   "field1="& field1 &
   "&field2="& field2 &
   "&secret=A12474&Submit=Submit%20for%20Processing")]
```

This will post the form data to the URL specified and return the result as raw HTML data. If the server does not react the function will time-out after gTimeOutTime ticks and an error code of "$$-4230" will be returned.

How to handle radio buttons and checkboxes in a form

With the URL plug-in you can also post forms that have radio buttons or checkboxes in it.

Say you want to fill in data of a web page. The web page has this FORM:

```
<HTML>
<HEAD><TITLE>Test Form</TITLE></HEAD>
<BODY>
<P>TEST</P>
<FORM ACTION="/cgi-sys/formmail.pl" METHOD=POST>
   <P><INPUT TYPE=checkbox NAME=checkbox1 VALUE=value1 CHECKED>checklabel</P>
   <P><INPUT TYPE=radio NAME=myradio VALUE=radiovalue2>radiolabel 1
   <INPUT TYPE=radio NAME=myradio VALUE=radiovalue2>radiolabel 2</P>
   <P>input data: <INPUT TYPE=text NAME=field1 VALUE="" SIZE=32><BR>
   <BR>
   <INPUT TYPE=submit NAME=Submit VALUE="Submit for Processing"></P>
</FORM>
</BODY>
</HTML>
```

This FORM has a checkbox, 2 radio buttons and an input field.

Checkboxes

In the POST data you only include the checkbox if it is checked. Otherwise leave it out.
When you add it, add the name, in this case "checkbox1", an equals sign and the value, here "value1".

So in the case above you add: "checkbox1=value1" for this checked box.

Radio buttons

For a radio button you include the value of the radio button that is selected.
In this example "myradio", an equals sign and the value of the chosen radio button, here "radiovalue2".

Putting it together you need to send this as the POST data (the 3rd parameter):

```
checkbox1=value1&myradio=radiovalue2&field1=typeddata&Submit=Submit+for+Processing
```

Pitfalls and considerations

- Javascripts may move entered data from one field in the web form to another just before the submit is done. Look if this happens in the javascript that's on the web page.

- Servers may use sessionIDs or other ways to make sure the web page is used only once. A sessionID may also time out. This may cause the submission of the web form to fail.

- Web site owners usually don't permit getting data from their web site in any way. Data is usually copyrighted, so be sure to check if you comply with the use of the data.

- Certain special characters and all "high ASCII" characters (characters with an ASCII value above 127) must be encoded when sent to a web server. The function TURL_ToHTTP of the plug-in returns the contents of the specified field or text value encoded in HTTP.

The special characters that must be encoded are
; / ? : @ = & > < # % { } ' | \ ^ ~ [ ] ` ¨ © " (space)
plus all high ASCII characters. These characters are encoded using the formula %nn where nn is the hexadecimal value that represents the character in the International Standards Organization (ISO) Latin-1 character set. For example %20 is the encoded value for the space character.

- The submit field may sometimes be left out as not all web servers require it.

URL Lengths

The length of URLs may be limited: a lot of web servers don't support URLs longer than 2048 characters. Note that the plug-in has higher limits: the total length of an URL should be less than 32768 characters.

This has consequences for the use of a GET, POST, PUT or DELETE method.

Use GET and POST in HTML forms - which is better?

In HTML, one can specify two different submission methods for a form. The difference between "GET" (the default) and "POST" is primarily defined in terms of form data encoding. The GET method sends extra data as part of the URL, while a POST has a fixed URL and sends the data separately.

If the set-up of the receiving web server is a given, you need to use the method that is specified in the web source. However, if you want to send longer strings of text (more than 2048 characters) you should use the POST method to submit the form.

NB: The official recommendations say that "GET" should be used if and only if the form processing is idempotent, which typically means a pure query form. Generally it is advisable to do so. There are, however, problems related to long URLs and non-ASCII character repertoires which can make it necessary to use "POST" even for idempotent processing.

See http://www.cs.tut.fi/~jkorpela/forms/methods.html for a more in-depth explanation on these issues.

Plug-in limitations and known issues

Troi URL Plug-in 3 still has some limitations. Please be aware of the following:

- The maximum size of an URL is 32768 characters, but see also the section above on URL lengths.
- You can not save data directly to disk. A possible solution is storing the data in a field and then use the Troi File Plug-in (see http://www.troi.com/software/fileplugin.html) to write it to disk.
- Error codes on Mac and on Windows can be different in the same situation.
- It is possible to use Troi URL Plug-in to access a file hosted with the Web Companion. However it won't work always if you are trying to access a URL on the same computer. This is the case if you use the same FileMaker Pro Client for the Web Companion and Troi URL Plug-in.

Cookies

A cookie is a piece of text that a Web server can store on a user's hard disk. Cookies allow a Web server to store information on a user's machine and later retrieve it. These cookies serve as state information, for example as an unique visitorID.

Cookies are supported by Troi URL Plug-in. On Windows cookies are handled natively by the operating system, meaning that cookies are sent to the web server without the plug-in need to do something. On Mac OS X you need to use the function TURL_SetCookies to specify which cookie text to send.

Which cookies do you need to send?

Some web servers want to receive cookies before they return the wanted data. In this case you need to find out first which cookies to send. You can do this by returning the headers of a web page in the TURL_Get or TURL_Post functions. For example we get the home page of Amazon:

Set Field [result , TURL_Get("-ReturnHeader" ; "http://www.amazon.com/")]

This will result in something similar to this (simplified) page:

```
HTTP/1.1 200 OK
Date: Mon, 27 Dec 2010 16:05:33 GMT
Server: Stronghold
Set-Cookie: session-id=104-1127-69752; path=/; domain=.amazon.com; expires=Monday, 07-
Apr-2011 08:00:00 GMT
Set-Cookie: session-id-time=1049400; path=/; domain=.amazon.com; expires=Monday, 07-
Apr-2011 08:00:00 GMT
Set-Cookie: obidos_path_continue-shopping=continue-shopping-url=/subst/home/home.html/
104-123397-6952&continue-shopping-post-data=&continue-shopping-
description=generic.gateway.default; path=/; domain=.amazon.com
Connection: close
Content-Type: text/html
<html> ...
</hmtl>
```

Each line that starts with "Set-Cookie:" is a request of the web server to set a cookie on this computer. In this case there are 3 cookies being set. If you want to sent those 3 in the next request add this ScriptMaker Script step before the next TURL_Get or TURL_Post step:

```
Set Field [ gErrorCode, TURL_SetCookies( "-unused" ;
"session-id=104-1127-69752; session-id-time=1049400; obidos_path_continue-
shopping=continue-shopping-url=/subst/home/home.html/104-123397-6952&continue-shopping-
post-data=&continue-shopping-description=generic.gateway.default
"  )]
```

You need to do this on Mac OS X only. On Windows cookies are handled natively by the operating system. So this function does not do anything on this platform.
NB: To view cookies on Windows XP open the control panel "Internet Options". Then click on the Settings button in the section "Temporary Internet files" under the General tab. Click on the View Files button to see the cookie files.

See http://www.howstuffworks.com/cookie1.htm for an explanation on how cookies work.

Function Reference

TURL_Delete

Syntax TURL_Delete(switches ; theURL)

Sends request to delete an item at a specified place on the web server, using the DELETE method.

Parameters

switches	this determines the behaviour of the plug-in
theURL	the url of the server

Switches can be one or more of these:

-NoDialog	don't display a progress dialog
-TimeoutTicks=x	specify the connect timeout time in x ticks (1/60th of a second)
-ReturnHeader	include the header in the returned text (at the beginning)
-Portnumber=y	specify the port number to use
-DontAutoRedirect	the plug-in will not go to a redirected page but return the original page
-AllowAnyRootCertificate	(for HTTPS) allow root certificates from unrecognized certification authorities. Note that this is less secure
-AlwaysSendUserPassword	will directly send the username + password, even when this might not be necessary
-DontTryWithUserPassword	don't send username + password when the web server requests authorization

Returned result

If successful it returns the data of the URL. If unsuccessful it returns an error code starting with $$ and the error code. Possible error codes are:

$$-1	user cancelled
$$-4230	the connection timed out
$$-3242	this protocol is not supported (use only http and https)
$$-30776	authentication error, you need to supply a correct user name and password
$$-92	servername part of the url is too long
$$-2110	path part of the url is too long

Other errors may be returned, specifically errors in the range 300 to 599, which are HTTP Status Codes.

Special considerations

Not every web server supports the DELETE method.
Secure connections (HTTPS) using SSL are also supported.
If you don't specify a timeout, a default timeout of 15 seconds is used.

On OS X a progress dialog can be displayed. On this platform you can also cancel a long operation with an ESC or a Command-Period.

The length of the URL is limited, by both the plug-in and receiving web servers. A lot of web servers don't support URLs longer than 2048 characters. Note that the plug-in has higher limits of 32867 characters for the URL.

Example usage

TURL_Delete("-Unused" ; "http://www.example.com/data/user123/mydata.txt")

This will send the request to the web server to delete the file at the location /data/user123/mydata.txt.

TURL_Delete

Example 2

We assume that in your FileMaker file the following field is defined:
ImageName Text

The field ImageName should contain the name of the image, for example "img123.jpg".

In ScriptMaker add the following script steps:
Set Variable[$thebaseURL, "http://www.photos.com/images/"]
Set Field[resultText, TURL_Delete(" -noDialog" ; $thebaseURL & ImageName)]

This will ask the web server to delete the image at the specified location on the web server: "http://www.photos.com/images/img123.jpg". Note that web servers might not support the DELETE method or refuse this DELETE request.

TURL_Get

Syntax TURL_Get(switches ; theURL)

Gets the raw data of the specified URL. This can be for example the HTML of a web page.

Parameters

switches	this determines the behaviour of the plug-in
theURL	the url to get

Switches can be one or more of these:

-NoDialog	don't display a progress dialog
-TimeoutTicks=x	specify the timeout time in x ticks (1/60th of a second)
-ReturnHeader	include the header in the returned text (at the beginning)
-ReturnDataAfterError	return the error code followed by the data
-Portnumber=y	specify the port number to use
-DontAutoRedirect	the plug-in will not go to a redirected page but return the original page
-AllowAnyRootCertificate	(for HTTPS) allow root certificates from unrecognized certification authorities. Note that this is less secure
-AlwaysSendUserPassword	directly send the username + password, even when this might not be necessary
-DontTryWithUserPassword	don't send username + password when the web server requests authorization
-ExtraImageCheck	the plug-in will perform extra tests to determine if it is a known image type
-DontDetectEncoding	disable automatic UTF-8 encoding detection, instead keep the native encoding
-Encoding=UTF8	forces the resulting webpage to be interpreted as UTF-8

Returned result

If successful it returns the data of the URL. If unsuccessful it returns an error code starting with $$ and the error code. Possible error codes are:

$$-1	user cancelled
$$-4230	the connection timed out
$$-3242	this protocol is not supported (use only http and https)
$$-30776	authentication error, you need to supply a correct user name and password
$$-92	servername part of the url is too long
$$-2110	path part of the url is too long

Other errors may be returned, specifically errors in the range 300 to 599, which are HTTP Status Codes.

Special considerations

Secure connections (HTTPS) using SSL are also supported on Windows and OS X.

If you don't specify a timeout, a default timeout of 15 seconds is used.

On OS X a progress dialog is displayed. On this platform you can also cancel a long operation with an ESC or a Command-Period.

The maximum length of the URL is limited, by both the plug-in and receiving servers. A lot of web servers don't support URLs longer than 2048 characters. Note that the plug-in has higher limits of 32867 characters. See also the user guide for more info on these limits.

On OS X you may need to set cookies to get the desired result. See the TURL_SetCookies function.

Some images on the web don't have a clear extension, or even have an extension GIF even if they are JPEGs! When using the switch -ExtraImageCheck the plug-in will perform extra tests to see if it is a known image type. If it is an image type, the plug-in returns the data as an image instead of the raw data as text.

Starting with v3.5 the plug-in can automatically detect UTF-8 encoded web pages (and encode them correctly). This is enabled by default, and is determined by looking for the meta and charset tag in the HTML result of the TURL_Get

function. For an XML result the "encoding=utf-8" tag will also be detected.

Example usage

TURL_Get("-unused" ; "http://www.example.com/")

This will return the specified web page. It will return something similar to:

<HTML><HEAD><TITLE>Example Web Page</TITLE></HEAD>
<BODY> You have reached this web page by typing www.example.com. These domain names are reserved for use in documentation and are not available for registration. </BODY>/HTML>

Example 2

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

theURL	Text
gTimeOutTime	Global, number

theURL should contain the URL, for example "http://www.filemaker.com". In ScriptMaker add the following script step:

```
Set Field[resultText, TURL_Get( "-NoDialog -TimeoutTicks="& gTimeOutTime& ; theURL) ]
```

This will get the get data of the URL (the raw web page), without showing a dialog. If the server does not react the function will timeout after gTimeOutTime ticks and an error code of "$$-4230" will be returned.

TURL_GetAuthorizationURL

Syntax TURL_GetAuthorizationURL(switches ; method ; requestTokenURL ; authorizationURLbegin)

Talks with a web service to get an authorization URL. With the returned URL the user can authorize the plug-in (in a web browser) to use this service.

Parameters

switches	determine the behaviour of the function
method	either POST or GET
requestTokenURL	URL to obtain a request token from the web service
authorizationURLbegin	first part of the authorization URL, which will be used to construct the complete URL

switches must be set to:
-OAuthProtocol

Returned result

If successful this function returns the complete authorization URL. With this URL the user can authorize the plug-in to use this service.

If unsuccessful it returns an error code starting with $$ followed by the error code.

Special considerations

This function provides the first step to authorize the plug-in.
The plug-in currently only supports the OAuth 1.0A Authorization protocol.

Example usage

```
Set Variable [ $Twitter_Request_token_step_URL; "https://api.twitter.com/oauth/request_token" ]
Set Variable [ $Twitter_Authorization_URL_Begin; "https://api.twitter.com/oauth/authorize" ]
#
#Get the URL to the authorization web page where the user can grant access:
Set Variable [ $Twitter_Authorization_URL;
TURL_GetAuthorizationURL( "-OAuthProtocol:" ; "POST" ; $Twitter_Request_token_step_URL ;
$Twitter_Authorization_URL_Begin) ]
Open URL [ $Twitter_Authorization_URL ]
```

TURL_GetLastHTTPStatusCode

Syntax TURL_GetLastHTTPStatusCode

Returns the HTTP response status code from the last executed HTTP request.

Parameters

none

Returned result

returns the HTTP status code from the last GET, POST, PUT or DELETE function you performed.
If there is no status code the result is 0.

Special considerations

See the appendix in the user guide for possible HTTP status codes.

Example usage

Set Variable [result, GetLastHTTPStatusCode] will for example return 200. This means the last request was handled OK.

Example 2

Set Field [thePage , TURL_Get("-unused" ; "http://www.example.com")]
Set Field [gStatusCode , TURL_GetLastHTTPStatusCode]

This will get the web page into the field thePage and put the HTTP status code into the global field gStatusCode.

TURL_GetLastProperties

Syntax TURL_GetLastProperties(switches)

Gets the (image) properties of the last retrieved image by a TURL_Get or a TURL_Post action.

Parameters

switches determines the behaviour of the function

switches can be one of these:
-ImageType get the image type actually returned, for example GIF
-ImageWidth get the width of the last image, in pixels
-ImageHeight get the height of the last image, in pixels

Returned result

Possible values for image types are currently:
UNKNOWN
JPEG
GIF
PNG

If the last GET or POST did not return an image UNKNOWN will be returned as image type and 0, 0 is returned as width and height.

Special considerations

Some web pages have images which have the wrong extension, for example abc.jpg, which is then actually a GIF. This function makes it possible to detect this.

Example usage

TURL_GetLastProperties("-ImageType") will for example return "JPEG".

Example 2

```
Set Field [ containerField , TURL_Get( "-unused" ; "http://www.troi.com/ima/sm_peter.jpg" )]
Set Field [ imageType , TURL_GetLastProperties( "-ImageType" ) ]
Set Field [ imageWidth , TURL_GetLastProperties( "-ImageWidth" ) ]
Set Field [ imageHeight , TURL_GetLastProperties( "-ImageHeight" ) ]
```

This will get an image and put it into a containerField and fill the other fields with JPEG, 48 and 48 respectively.

TURL_HMACSHA1

Syntax TURL_HMACSHA1(switches ; key ; messageText)

Calculate a keyed-hash message authentication code (HMAC-SHA1) signature using a (secret) key.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
key	the secret key to sign the text with
messageText	the text of the message you want to sign

Returned result

Returns the HMAC-SHA1 signature

Special considerations

See http://en.wikipedia.org/wiki/HMAC for more information on this signature.

Example usage

```
Set Variable[ $signing_key ; "ABC12345676890" ]
Set Variable[ $messageString  ; "Some text to sign" ]
Set Variable[ $result_HMACSHA1_Signature ; TURL_HMACSHA1( "-Unused" ; $signing_key ; $messageString ) ]
```

The result will be the signature: "NUHabjnddiy6+/rC2VMOB7cbLrg="

TURL_IsSecure

Syntax TURL_IsSecure(switches)

Indicates if the last request was secure.

Parameters

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

Returned result

If the last request was secure this function returns 1.
If the last request was NOT secure this function returns 0.

TURL_Post

Syntax TURL_Post(switches ; theURL ; theData)

Sends data to the server, using the POST method, and returns the result as raw data.

Parameters

switches	this determines the behaviour of the plug-in
theURL	the url of the server
theData	the data to be sent with the POST method

Switches can be one or more of these:

-NoDialog	don't display a progress dialog
-TimeoutTicks=x	specify the connect timeout time in x ticks (1/60th of a second)
-ReturnHeader	include the header in the returned text (at the beginning)
-ReturnDataAfterError	return the error code followed by the data
-Portnumber=y	specify the port number to use
-NotEncoded	send the post data without any normal encoding for "www-form-urlencoded"
-DontAutoRedirect	the plug-in will not go to a redirected page but return the original page
-AllowAnyRootCertificate	(for HTTPS) allow root certificates from unrecognized certification authorities. Note that this is less secure
-ExtraCRLFafterData	an extra CRLF (carriage return and line feed) character is added at the end of the Post data
-AlwaysSendUserPassword	will directly send the username + password, even when this might not be necessary
-DontTryWithUserPassword	don't send username + password when the web server requests authorization
-Encoding=ISO_8859_1	encodes the data in ISO-8859-1 (Latin-1)
-Encoding=UTF8	encodes the data in UTF-8, and also the resulting data
-DontDetectEncoding	disable automatic UTF-8 encoding detection, instead keep the native encoding

Returned result

If successful it returns the data of the URL. If unsuccessful it returns an error code starting with $$ and the error code. Possible error codes are:

$$-1	user cancelled
$$-4230	the connection timed out
$$-3242	this protocol is not supported (use only http and https)
$$-30776	authentication error, you need to supply a correct user name and password
$$-92	servername part of the url is too long
$$-2110	path part of the url is too long

Other errors may be returned, specifically errors in the range 300 to 599, which are HTTP Status Codes.

Special considerations

Secure connections (HTTPS) using SSL are also supported on Windows and OS X.
If you don't specify a timeout, a default timeout of 15 seconds is used.
See the user guide section "How to make sense of a web form", to determine what you must send to make the form work.

On OS X a progress dialog is displayed. On this platform you can also cancel a long operation with an ESC or a Command-Period.

The length of the URL is limited, by both the plug-in and receiving servers. A lot of web servers don't support URLs longer than 2048 characters. Note that the plug-in has higher limits of 32867 characters for the URL. The 3rd parameter "theData" has a limit of about a half GB of characters.
On OS X you might need to set cookies to get the desired result. See the TURL_SetCookies function.
See also TURL_SetCustomHeader for special headers, that may be needed, for example in a form.

Starting with v3.5 the plug-in can now automatically detect UTF-8 encoded web pages (and encode them correctly). This

is enabled by default, and is determined by looking for the meta and charset tag in the HTML result of the TURL_Post function. For an XML result the "encoding=utf-8" tag will also be detected.

Example usage

TURL_Post("-unused" ; "http://www.idninc.com/cgi-bin/sherlock.cgi" ; "name=troi")

This will send the form data "name=troi" for the Sherlock form "cgi-bin/sherlock.cgi" to the server www.idninc.com and return the result. This will return something similar to:

```
<HTML>
<HEAD><TITLE>COMPANY SEARCH</TITLE></HEAD>
<body><CENTER><!--BANNER_START--><A HREF="http://www.BestSiteFirst.com">
<IMG SRC="http://www.idninc.com/images/banner.gif" border=0></A>
<!--BANNER_END-->
...
<!--SEARCH_RESULTS_START-->
<P> <!--RELEVANCE-->100<!--END_RELEVANCE-->
% <A HREF="http://www.troi.com">Troi Automatisering: For FileMaker Pro plug-ins</A>
<UL>Check out the latest FileMaker Pro Plug-ins, Tips and Shareware from Troi Automatisering  </UL></P>
...
<!--SEARCH_RESULTS_END-->
</UL></BODY>
</HTML>
```

Example 2

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

theURL	Text
gTimeOutTime	Global, number

theURL should contain the URL, for example "http://www.filemaker.com". In ScriptMaker add the following script steps:

```
Set Field[formData,  "name=troi&number=123456"]
Set Field[resultText, TURL_Post(" -TimeoutTicks=" & gTimeOutTime ; theURL ; formData) ]
```

This will post the form data to the URL specified and return the result as raw HTML data. If the server does not react the function will timeout after gTimeOutTime ticks and an error code of "$$-4230" will be returned.

TURL_Put

Syntax TURL_Put(switches ; theURL ; theData)

Sends data to the server or transfers a file to the server, using the PUT method.

Parameters

switches	this determines the behaviour of the plug-in
theURL	the url of the server
theData	the data to be sent with the PUT method. This can also be a file or image in a container field

Switches can be one or more of these:

-NoDialog	don't display a progress dialog
-TimeoutTicks=x	specify the connect timeout time in x ticks (1/60th of a second)
-ReturnHeader	include the header in the returned text (at the beginning)
-ReturnDataAfterError	return the error code followed by the data
-Portnumber=y	specify the port number to use
-DontAutoRedirect	the plug-in will not go to a redirected page but return the original page
-AllowAnyRootCertificate	(for HTTPS) allow root certificates from unrecognized certification authorities. Note that this is less secure
-AlwaysSendUserPassword	will directly send the username + password, even when this might not be necessary
-DontTryWithUserPassword	don't send username + password when the web server requests authorization

Returned result

If successful it returns the data of the URL. If unsuccessful it returns an error code starting with $$ and the error code. Possible error codes are:

$$-1	user cancelled
$$-4230	the connection timed out
$$-3242	this protocol is not supported (use only http and https)
$$-30776	authentication error, you need to supply a correct user name and password
$$-92	servername part of the url is too long
$$-2110	path part of the url is too long

Other errors may be returned, specifically errors in the range 300 to 599, which are HTTP Status Codes.

Special considerations

Not every web server supports the PUT method.
Note that theData is UTF8 Encoded by default.
Secure connections (HTTPS) using SSL are also supported.
If you don't specify a timeout, a default timeout of 15 seconds is used.

On OS X a progress dialog is displayed. On this platform you can also cancel a long operation with an ESC or a Command-Period.

The length of the URL is limited, by both the plug-in and receiving web servers. A lot of web servers don't support URLs longer than 2048 characters. Note that the plug-in has higher limits of 32867 characters for the URL.

Example usage

TURL_Put("-Unused" ; "http://www.example.com/data/user123/mydata.txt" ; "color=blue")

This will send the data "color=blue" to be put at the location /data/user123/mydata.txt on the web server.

Example 2

TURL_Put

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

MyContainer	Container
ImageName	Text

The field MyContainer can contain a file or an image. ImageName should contain the name of the image, for example "img123.jpg".

Add the following script steps:

```
Set Variable[ $thebaseURL, "http://www.photos.com/images/"]
Set Field[resultText, TURL_Put(" -noDialog" ; $thebaseURL & ImageName ; MyContainer) ]
```

This will transfer the imagefile in the container to the web server, to be put in the specified location on the web server: "http://www.photos.com/images/img123.jpg". Note that web servers might not support the PUT method or refuse this PUT request.

TURL_Reinitialize

Syntax TURL_Reinitialize(switches)

Restarts the plug-in.

Parameters

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

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

At the moment this function does not return errors.

Special considerations

Specifically the proxy settings will be read again.

TURL_SendAuthorizationPIN

Syntax TURL_SendAuthorizationPIN(switches ; method ; accessTokenURL ; PIN)

This will send the PIN (obtained while granting access) to the web service to complete the authorization.

Parameters

switches	determine the behaviour of the function
method	either POST or GET
accessTokenURL	URL to obtain the final access token (and access token secret) from the web service
PIN	The PIN (obtained by the user while granting access)

switches must be set to:
-OAuthProtocol

Returned result

If successful this function returns several named parameters separated by ampersands, in this result are the two credentials access_token and access_token_secret

Special considerations

This function provides the second step and completes the authorization.
The plug-in currently only supports the OAuth 1.0A Authorization protocol.

The credentials access_token and access_token_secret need to be stored, to be able to perform requests in the future. Normally you can keep using these two credentials, until the user revokes them (on the web service site). Take care to keep the credentials secret, as they are tied to the userID on the web service.

Example usage

```
Set Variable [ $PIN; "123456" ]
Set Variable [ $Twitter_Access_token_step_URL , "https://api.twitter.com/oauth/access_token" ]
#send the PIN to Twitter to complete the authorization:
Set Variable [ $Twitter_Result; Value:TURL_SendAuthorizationPIN( $Twitter_Access_token_step_URL ; $PIN ) ]
```

if succesfull $Twitter_Result will be like:

```
oauth_token=123456-1w4FgHt&oauth_token_secret=SN4M6F99GG&user_id=12345&screen_name=troi
```

See the example on how to parse this into separate fields.

TURL_SendAuthorizedRequest

Syntax TURL_SendAuthorizedRequest(switches ; method ; requestURL; accessToken ; accessTokenSecret ; theRequest)

Sends a request (like posting a tweet), using the 2 credentials obtained during authorization.

Parameters

switches	determine the behaviour of the function
method	either POST or GET
requestURL	URL of the web service to where the request must be sent
accessToken	part 1 of the 2 credentials obtained during authorization
accessTokenSecret	part 2 of the 2 credentials obtained during authorization
theRequest	the request you want to perform

switches must be set to:
-OAuthProtocol

Returned result

If successful this function returns the result of the request. If unsuccessful it returns an error code starting with $$ followed by the error code.

Special considerations

With this function you can perform the actual work by sending requests to the web service, for example post a tweet on Twitter.
You need to have gotten credentials first with the two authorization functions TURL_GetAuthorizationURL and TURL_SendAuthorizationPIN.

The plug-in currently only supports the OAuth 1.0A Authorization protocol.

Example usage

```
#This script will post a tweet on Twitter.
#Get the text to tweet in a variable:
Set Variable [ $TweetText; this::TweetText ]
#Set up the other variables:
Set Variable [ $URL; "http://api.twitter.com/1/statuses/update.json" ]
#Iuse the credentials stored in two global fields:
Set Variable [ $oauth_token; this::gOAuth_token ]
Set Variable [ $oauth_token_secret; this::gOAuth_token_secret ]
#Build the request to tweet (= a status update in the API):
#NOTE: the text of the tweet must be URL Encoded.
Set Variable [ $request; "status=" & TURL_ToURLEncoded( "-Unused" ; this::TweetText ) ]
#
#Now send the tweet to Twitter, using the POST method, and the oauth credentials.
Set Variable [ $Result; TURL_SendAuthorizedRequest( "-OAuthProtocol:" ; "POST" ; $URL; $oauth_token ;
$oauth_token_secret ; $request) ]
Set Field [ this::Received text; $Result ]
```

TURL_SetAuthCredentials

Syntax TURL_SetAuthCredentials(switches ; consumerKey ; consumerSecret)

Sets the Consumer Key and Consumer Secret for an OAuth authorization.

Parameters

switches	this determines the behavior of the plug-in
consumerKey	the consumer key of your solution
consumerSecret	the consumer secret of your solution

Returned result

The returned result is an error code. If the consumerKey and the consumerSecret have been set successfully the plug-in 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

Use this function to set up an OAuth connection to a web service.
Setting these credentials makes it possible to work with more web services who use OAuth 1.x.
See the OAuthWebService example file for more information on the needed steps.

Example usage

TURL_SetAuthCredentials("-Unused" ; "CKVA0123456789" ; "CSABC123445566778")

TURL_SetCookies

Syntax TURL_SetCookies(switches ; cookiesText)

Sets the cookies (on OS X) to be sent as part of a GET or POST request.

Parameters

switches	not used, reserved for future use. Leave blank or put "-unused"
cookiesText	the text of the cookies. Separate each cookie with a semicolon

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code. Possible error codes are:

0	no error
$$-92	length of the cookies text is too long
$$-4221	this function does not do anything on Windows

Special considerations

The cookie text is remembered until you specify a different cookie text. If you want to remove a previously specified cookie text, call this function with an empty cookie text.
The cookie text is also removed when you quit FileMaker Pro. The maximum length of a cookie text is 1 Gb.

To see which cookies are set by a web server add the switch "-ReturnHeader" to the TURL_Get or TURL_Post functions of the web page.

On Windows cookies are handled natively by the operating system. So this function does not do anything on this platform. Note: to view cookies on Windows open the control panel "Internet Options". Then click on the Settings button in the section "Temporary Internet files" under the General tab. Click on the View Files button to see the cookie files.

See also http://www.howstuffworks.com/cookie1.htm for an explanation on how cookies work.

Example usage

Set Field [result, TURL_SetCookies("-unused " ; "troicookie=aap ; plunk=A32")]

This will send the 2 cookies with each subsequent request.

Set Field [result, TURL_SetCookies("-unused" ; "")]

This will clear the cookie text. No cookies will be sent with a request.

Example 2

To find out which cookies to send we first return the headers of a web page in the TURL_Get or TURL_Post functions. In this case we get the home page of Amazon:

Set Field [result , TURL_Get("-ReturnHeader" ; "http://www.amazon.com/")]

This will result in something similar to this (simplified) page:

HTTP/1.1 200 OK
Date: Mon, 31 Mar 2003 16:05:33 GMT
Server: Stronghold
Set-Cookie: session-id=104-1127-69752; path=/; domain=.amazon.com; expires=Monday, 07-Apr-2003 08:00:00 GMT
Set-Cookie: session-id-time=1049400; path=/; domain=.amazon.com; expires=Monday, 07-Apr-2003 08:00:00 GMT
Set-Cookie: obidos_path_continue-shopping=continue-shopping-url=/home.html/104-123397; path=/; domain=.amazon.com

TURL_SetCookies

Connection: close
Content-Type: text/html
<html> ...</hmtl>

Each line that starts with "Set-Cookie:" is a request of the web server to set a cookie on this computer. In this case there are 3 cookies being set. If you want to send those 3 in the next request add this Script step before the next GET or POST request:

```
Set Field [ gErrorCode, TURL_SetCookies("-unused" ;
            "session-id=104-1127-69752; session-id-time=1049400;
            obidos_path_continue-shopping=continue-shopping-url=/home.html/104-123397"  )]
```

TURL_SetCustomHeader

Syntax TURL_SetCustomHeader(switches ; customHeaderText)

Sets a custom header to be used.

Parameters

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

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

At the moment this function does not return errors.

Special considerations

Some web browsers require a special header text before they return the wanted data. This function allows you to set this text.

The custom header text is remembered until you specify a different text. The text is also removed when you quit FileMaker Pro. The maximum length of the custom header text is 1Gb.

If you don't specify a custom header text the plug-in will use a default header. If you specify an empty string the default header will be used.

Example usage

```
TURL_SetCustomHeader("-unused " ;
"Accept: */*
Content-Type:text/html" )
```

This will send the custom header with each subsequent request. If you want to reset to the default headers use:

```
Set Field [ result, TURL_SetCustomHeader( "-unused" ; "")]
```

This will clear the custom header text. No custom headers will be sent with a request.

Example 2

Some web servers expect web forms to be sent with specific custom headers (usually x-www-form-urlencoded). In this case you should set the headers before you send the form to the server. This can be done like this:

```
Set Variable [ $result ; TURL_SetCustomHeader( "-unused"  ; "Content-Type:application/x-www-form-urlencoded" ) ]
Set Variable [  $resultData, TURL_Post( "-unused" ; $URL ;  $FormData)]
```

Here $URL is a variable with the URL of the POST and $FormData is a variable that contains the form data to be sent.

TURL_SetPassword

Syntax TURL_SetPassword(switches ; password)

Sets the password to be used for password protected URLs.

Parameters

switches	not used, reserved for future use. Leave blank or put "-unused"
password	the password (this is case sensitive)

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

At the moment this function does not return errors.

Special considerations

The password is remembered until you specify a different password. If (for security) you want to remove a previously specified password, call this function with an empty password. The password is also removed when you quit FileMaker Pro.

The maximum length of a password is 1 Gb.

Example usage

TURL_SetPassword("-unused" ; "Als ik zo vrij mag zijn")

This will set the password to "Als ik zo vrij mag zijn".

Example 2

This will set the user name and password and get a web page. We assume that in your FileMaker file the following fields are defined:

gUserName	Global, text
gPassword	Global, text

gUserName should contain the user name, for example "Olie B. Bommel" and gPassword should contain the password, for example "denkraam". Add the following script steps:

```
Set Field[gErrorCode, TURL_SetUserName( "-unused" ;  gUserName) ]
If [gErrorCode = 0]
   Set Field[gErrorCode, TURL_SetPassword( "-unused" ; gPassword) ]
End if
If [gErrorCode = 0]
   Set Field[gResult, TURL_Get( "-NoDialog" ; "http://www.mtvc.nl/") ]
   # forget the user name and password
   Set Field[gErrorCode, TURL_SetUserName( "-unused" ;  "") ]
   Set Field[gErrorCode, TURL_SetPassword( "-unused" ;  "") ]
End if
```

TURL_SetProgressText

Syntax TURL_SetProgressText(switches ; messageText ; buttonText)

Sets a custom text to be used for the progress dialog.

Parameters

switches	not used, reserved for future use. Leave blank or put "-unused"
messageText	the text of the progress dialog
buttonText	(optional) the text of the stop button

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

At the moment this function does not return errors.

Special considerations

The text is remembered until you specify a different text. The text is also removed when you quit FileMaker Pro.

The maximum length of the message text is 255 characters. The maximum length of the button text is 20 characters.

At the moment the plug-in shows a Progress Dialog only on OS X.

Example usage

TURL_SetProgressText("-unused" ; "Getting your data..." ; "Stop")

Example 2

This will get a web page with a custom text. We assume that in your FileMaker file the following fields are defined:

urlField	text
gErrorCode	Global, text

urlField should contain the URL you are getting, for example "www.troi.com" . Add the following script steps:

```
Set Field[gErrorCode, TURL_SetProgressText("-unused" ; "Getting data from: " & urlField ; "Halt") ]
Set Field[resultText, TURL_Get("-unused " ; urlField) ]
```

TURL_SetProxy

Syntax TURL_SetProxy(switches ; proxyAddress ; proxyPortNr ; proxyUserName; proxyPassword)

Sets an explicit proxy server to be used, via which the data is sent to the web server.

Parameters

switches	not used, reserved for future use. Leave blank or put "-unused"
proxyAddress	the IP address of the proxy server
proxyPortNr	(optional) the portnr of the proxy server
proxyUserName	(optional) the user name to be sent as authentication to the proxy server
proxyPassword	(optional) the password to be sent as authentication to the proxy server

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

At the moment this function does not return errors.

Special considerations

If you don't use this function the system settings for proxies will be used.
You can reset the proxy by specifying an empty proxyAddress string. The systems settings will be used then.

Some proxy servers require authentication, which you can supply with the optional proxyUserName and proxyPassword parameters.

Example usage

TURL_SetProxy("-unused"; "proxy.server.com" ; 1234)

This sets the proxy to the server at proxy.server.com at port 1234.

Example 2

Set Field [gErrorCode , TURL_SetProxy("-unused"; "192.168.0.60" ; 8000)]

This sets the proxy to the server at address "192.168.0.60" and port 8000.

Set Field [gErrorCode , TURL_SetProxy("-unused"; "")]

This resets the proxy setting. The plug-in will use the system settings for proxies.

TURL_SetUserAgent

Syntax TURL_SetUserAgent(switches ; userAgentText)

Sets a custom text to be used for the name of the user agent requesting the information. The user agent can be set to the name of a popular browser, so that unwanted redirects don't occur.

Parameters

switches	not used, reserved for future use. Leave blank or put "-unused"
userAgentText	the user agent text

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

At the moment this function does not return errors.

Special considerations

The text is remembered until you specify a different text. The text is also removed when you quit FileMaker Pro. The maximum length of the userAgentText is 1Gb.

If you don't specify a user agent text the plug-in will use a default of "Troi URL Plug-In 4.5 (Windows)" or "Troi URL Plug-In 4.5 (Macintosh; Mach-O)" respectively.
If you specify an empty string the default text will be used.

Example usage

TURL_SetUserAgent("-unused"; "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; Trident/6.0)")

This pretends this is Internet Explorer 10 on Windows.

TURL_SetUserAgent("-unused"; "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/601.6.17 (KHTML, like Gecko) Version/9.1.1 Safari/601.6.17")

This pretends this is Safari on OS X.

Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; Trident/6.0)

TURL_SetUserName

Syntax TURL_SetUserName(switches ; username)

Sets the user name to be used for password protected URLs.

Parameters

switches	not used, reserved for future use. Leave blank or put "-unused"
username	the user name (this is case sensitive)

Returned result

If successful this function returns 0. If unsuccessful it returns an error code starting with $$ followed by the error code.

At the moment this function does not return errors.

Special considerations

The user name is remembered until you specify a different user name. If (for security) you want to remove a previously specified user name, call this function with an empty user name.
The user name is also removed when you quit FileMaker Pro.

The maximum length of a username is 255 characters.

Example usage

TURL_SetUserName("-unused" ; "Tom Poes")

This will set the user name to "Tom Poes".

Example 2

This will set the user name and password and get a web page. We assume that in your FileMaker file the following fields are defined:

gUserName	Global, text
gPassword	Global, text

gUserName should contain the user name, for example "Olie B. Bommel" and gPassword should contain the password, for example "denkraam". Add the following script steps:

```
Set Field[gErrorCode, TURL_SetUserName( "-unused" ;  gUserName) ]
If [gErrorCode = 0]
   Set Field[gErrorCode, TURL_SetPassword( "-unused" ;  gPassword) ]
End if
If [gErrorCode = 0]
   Set Field[gResult, TURL_Get( "-NoDialog" ; "http://www.mtvc.nl/") ]
   # forget the user name and password
   Set Field[gErrorCode, TURL_SetUserName( "-unused" ;  "") ]
   Set Field[gErrorCode, TURL_SetPassword( "-unused" ;  "") ]
End if
```

TURL_ToHTTP

Syntax TURL_ToHTTP(switches ; text)

Encodes a text in HTTP.

Parameters

switches determine the behavior of the function
text either the name of a text field or a text constant (in quotes) that you want converted to HTTP

You can leave switches empty or add this switch:
-UpperCaseHex make the encoded string contain only uppercase hexadecimals, like for example %0A (instead of %0a) for the linefeed character.

Returned result

The result is the HTTP encoded text. Certain special characters and all "high ASCII" characters (characters with an ASCII value above 127) must be encoded when converted to HTTP.

TURL_ToHTTP returns the contents of the specified field or text value encoded in HTTP. The special characters that must be encoded are:
; / ? : @ = & > < # % { } ' | \ ^ ~ [ ] ` ¨ © " (space)
plus all high ASCII characters. These characters are encoded using the formula %nn where nn is the hexadecimal value that represents the character in the International Standards Organization (ISO) Latin-1 character set. For example, %20 is the encoded value for the space character.

Special considerations

This function also runs in a runtime created with FileMaker Pro Advanced.

NOTE Use the TURL_ToURLEncoded function if you want to encode higher Unicode characters too.

Example usage

Set Field [result, TURL_ToHTTP("-unused" ; "Hello World!")]

gives this result: "Hello%20World!"

Example 2

In a database you have defined a text field named theURL which contains "<http://www.filemaker.com>". Create a script with this step:

Set Field [result, TURL_ToHTTP("-unused" ; theURL)]

Then the result will contain the encoded text:

%3Chttp%3A%2F%2Fwww.filemaker.com%3E

TURL_ToURLEncoded

Syntax TURL_ToURLEncoded(switches ; text)

URL encodes text (also known as percent-encoding).

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
text	the text to encode

Returned result

Returns the URL Encoded text

Special considerations

Note that the text is first converted to UTF-8 bytes, which makes it possible for higher Unicode characters to be encoded too.

See also http://en.wikipedia.org/wiki/Percent-encoding.

Example usage

TURL_ToURLEncoded("-Unused" ; "Hello World!")

this will give as result:
"Hello%20World%21"

TURL_Version

Syntax TURL_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 these:

-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

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 URL Plug-in 2.5".

VersionNumber:
If you asked for the version number it returns the version number of the plug-in x1000. For example version 2.4 will return number 2400.

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

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 a question mark to any external function that is not loaded.

Example usage

TURL_Version("") will for example return "Troi URL Plug-in 2.0.3".

Example 2

TURL_Version("-GetVersionNumber") will return 2030 for version 2.0.3.
TURL_Version("-GetVersionNumber") will return 4530 for (a future) version 4.5.3

So if you want to test for a feature introduced with version 2.5, test if the result is equal or greater than 2500.

TURL_VersionAutoUpdate

Syntax TURL_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 TURL_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

TURL_VersionAutoUpdate will return 04050000 for version 4.5
TURL_VersionAutoUpdate will return 04050203 for version 4.5.2.3

So for example to use a feature introduced with version 4.5 test if the result is equal or greater than 04050000.

HTTP Status Codes

Communicating with a web server is done with the help of Status Codes. Below you find the possible codes. Some of those may be returned by our plug-in.

Informational 1xx

no.	what
100	Continue
101	Switching Protocols

Successful 2xx

no.	what
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
204	No Content
205	Reset Content
206	Partial Content

Redirection 3xx

no.	what
300	Multiple Choices
301	Moved Permanently
302	Moved Temporarily
303	See Other
304	Not Modified
305	Use Proxy
306	(Unused)
307	Temporary Redirect

Client Error 4xx

no.	what	description
400	Bad Request	The request could not be understood by the server due to incorrect syntax.
401	Unauthorized User	Authentication is required.
402	Payment Required	
403	Forbidden	The server understood the request, but is refusing to fulfill it.
404	Page Not Found	The server has not found anything matching the Request-URI.
405	Method Not Allowed	The method specified in the Request-Line is not allowed for the resource identified by the Request-URI.
406	Not Acceptable	The server cannot generate a response that the requestor is willing to accept.
407	Proxy Authentication Required	This code is similar to 401 (Unauthorized), but indicates that the client must first authenticate itself with the proxy.
408	Request Timed Out	The server stopped waiting for a client request.
409	Conflict	The request could not be completed due to a conflict with the current state of the resource.
410	Gone	The requested resource is no longer available at the server and no forwarding address is known. This condition is similar to 404, except that the 410 error condition is expected to be permanent.
411	Length Required	The server requires a content-length in the request.
412	Precondition Failed	The precondition given in one or more of the request-header fields evaluated to false when it was tested on the server.
413	Request Entity Too Large	The server is refusing to process a request because the request entity is larger than the server is willing or able to process.
414	Request URL Too Long	The server is refusing to service the request because the Request-URL is longer than the server is willing to interpret.
415	Unsupported Media Type	The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.
416	Requested Range Not Satisfiable	
417	Expectation Failed	

Server Error 5xx

no.	what	description
500	Internal Server Error	Internal Web server error
501	Not Implemented	Function not implemented in Web server software
502	Bad Gateway	Bad Gateway; a server being used by this Web server has sent an invalid response.
503	Service Unavailable	Service unavailable because of temporary overload or maintenance.
504	Gateway Timeout	A server being used by this server has not responded in time.
505	HTTP Version Not Supported	The server does not support the HTTP protocol version that was used in the request message.