Error custom functions & error source management

Overview

This page describes the intended implementation of error management within a FileMaker solution. This page outlines a method of error handling which extends far beyond using FileMaker's own built-in Get (LastError) function. Your FileMaker solution can potentially experience far more errors than just those generated by FileMaker itself.

The temporary GitHub repository contains the custom functions mentioned on this page, and a sample FileMaker 12 file. Once the functions are complete, they will be moved to the main fmpstandards repository.

Error Origination

There are many places where an error can occur within a FileMaker solution. It doesn't stop with just FileMaker the application. The most fundamental errors within FileMaker are obviously those generated by FileMaker itself. These FileMaker error codes could be handled with FileMaker's own Get (LastError) function. However, there are many external technologies (such as plugins), as well as the logic used to create your FileMaker solution which can also generate errors. We refer to the origin of an error as an error type. The following items are the suggested classifications for error types.

<table>
<thead>
<tr>
<th>type</th>
<th>explanation of this type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fmp</td>
<td>FileMaker error code</td>
</tr>
<tr>
<td>App</td>
<td>An error which the developer generates within the logic of your own solution. An example of this would be a Perform script step higher in a sequence of steps which is required in order for subsequent Perform script steps to function properly.</td>
</tr>
<tr>
<td>Plugin:</td>
<td>Name</td>
</tr>
<tr>
<td>Function:</td>
<td>Name</td>
</tr>
<tr>
<td>other</td>
<td>Any error source can have its own error type associated with it; here are some examples:</td>
</tr>
<tr>
<td></td>
<td>• if you are interacting with a SOAP web service, the error type could be the name of the web service.</td>
</tr>
<tr>
<td></td>
<td>• if writing a mFM Module, the error type could be the name of the module, like: “mFM: JSON”</td>
</tr>
</tbody>
</table>

Error Storage

Because any error may have many different causal facets, such as relevant associated environmental information at the time the error occurred, a predefined number of error attributes is unrealistic. One of the primary goals is to capture environment specific information at the time of the error. Due to the amount of possible supporting information around any given error, all information relating to the error should be saved to the reserved variable $error. This variable contains name/value pairs, encoded by the #(name;value) function and accessible by the #Get and #Assign functions.

example of $error variable contents

```
$errorType = "Fmp";
$errorCode = GetAsNumber ( 105 ) ;
$errorDescription = "Layout is missing";
$errorInfo = "Go to some layout";
$AllowAbortState = GetAsNumber ( 1 );
$CurrentHostTimestamp = GetAsTimestamp ( "11/15/2013 5:36:12 PM" );
$errorCaptureState = GetAsNumber ( 0 );
$LastODBCError = "";
$ScriptName = "Run Error Examples";
$ScriptParameter = "";
$ScriptResult = "";
$UserCount = GetAsNumber ( 1 );
```

Custom Functions

Using a unique custom function for every unique source of an error provides a powerful method of capturing all necessary data about an error in a standardized format. All error specific functions are prefixed with the reserved domain of “Error”.
Error ( theErrorType ; theErrorCode ; theErrorDescription ; theErrorInfo )

Return Let Notation containing information about the error and the environment it occurred in.

ErrorFmp ( theErrorCode ; theErrorInfo )

Return Let Notation containing information about a FileMaker error and the environment it occurred in.

ErrorFmpGetLast ( theErrorInfo )

Return Let Notation containing information about the last FileMaker error and the environment it occurred in.

Produces the same result as ErrorFmp ( Get ( LastError ) ; "info about where the error occurred" ), but does not depend on ErrorFmp function.

ErrorApp ( theErrorCode ; theErrorInfo )

Return Let Notation containing information about the specified solution-specific error and the environment it occurred in.

ErrorFound ( theErrorData )

Return errorCode contained in theErrorData as a Boolean. In other words, if any error occurred at all, return True (1), otherwise return False (0).

Since $error is the reserved variable name for storing error data, this function will almost always be uses like this: ErrorFound ( $error )

Error[SOURCE] ( any ; parameters ; necessary ; theErrorInfo )

Where SOURCE is unique to the origination of the error and is the same as/similar to errorType. Examples of plugin error functions would be ErrorPlugInScriptMaster or ErrorPlugInTroiDialog. The code repository contains some examples of this type of function. The intention of these functions are to translate the error information provided by the SOURCE into the standardized format implemented by these custom functions.

Over time, it would be nice to see the community and plugin developers contribute error handling custom functions for various plugins and services.

Using the Custom Functions
### ErrorFmpGetLast

Go to Layout [Client: List]
Set Variable [$error; Value:ErrorFmpGetLast ( "go to Client: List layout" )]

Additional name/value pairs can be added to the $error variable using the # ( name ; value ) function:

### ErrorFmpGetLast with additional data

Set Field By Name [$fieldName; $value]
Set Variable [$error; Value:ErrorFmpGetLast ( "go to Client: List layout" ) & # ( "fieldName" ; $fieldName ) & # ( "value" ; $value )]

### ErrorApp

If [Get ( FoundCount ) = 0]
   Set Variable [$error; Value:ErrorApp ( 4 ; "validation section at start of script" )]
End If
# the parameter 4 is an example only, the code and description for the code must be defined in the ErrorApp custom function

### ErrorFound

If [ErrorFound ( $error )]
   # log the error
   Exit Script [Result: # ( "error" ; $error )]
End If