Custom function » Debug

While developing within FileMaker, there are many different situations in which critical information about the current state of the environment is helpful. In these cases, you can either use the many possible Get() functions provided by FileMaker, or you can combine these functions within one useful custom function named Debug(). The Debug() function used on this site is dependent on the Developer() custom function because you don't want to accidentally show debug information to a non-developer account.

Debug() function

```/**
 * ===================================================================================
 * Debug ( )
 * *
 * PARAMETERS: none
 * *
 * RETURNS: string) A list of informative values about the current FMP environment.
 * *
 * DEPENDENCIES: Developer ()
 * *
 * NOTES: *
 * *
 * RELEASE: 101009 - Initial by Matt Petrowsky
 * 2013-01-25 - Updated to be recursive and use better output. Included more functions
 * ===================================================================================
 */
If ( Developer;
Let ( [
    ~input = List ("Environment";
    "ApplicationVersion";
    "SystemPlatform";
    "SystemVersion";
    "ApplicationLanguage";
    "SystemLanguage";
    "HostName";
    "HostApplicationVersion";
    "SystemIPAddress";
    "HostIPAddress";
    "CurrentTimeStamp";
    "CurrentHostTimeStamp";
    "InstalledFMPlugins";
    "AllowAbortState";
    "ConnectionState";
    "CustomMenuSetName";
    "FileName";
    "FileSize";
    "FoundCount";
    "SystemNICAddress";
    "PersistentID";
    "RecordOpenCount";
    "Errors";
    "LastError";
    "LastODBCError";
    "User";
    "AccountName";
    "AccountPrivilegeSetName";
    "CurrentPrivilegeSetName";
    "AccountExtendedPrivileges";
    "CurrentExtendedPrivileges";
    "Layout";
    "LayoutName";
    "LayoutNumber";
    "LayoutTableName";
Extending Debug() output
In order to append additional information to the default information provided by the Debug function, you would simply use a List() function. For example, appending a local counter to the debug information.

```
List {
  Debug ();
  "Local counter $i = " & $i;
}
```

### Using Debug() within a layout

You can easily integrate Debug into the layout by using a button with a "bug" icon on it. By directly attaching a single script step of Show Custom Dialog, you can use the following code.

```
Let ( [
  $$DEBUG = Debug
];
  "Check the Data Viewer for debug information within the $$DEBUG variable. You can also put a "<<$$DEBUG>>" merge variable on your layout."
)
```

The Debug implementation is available within the Standards.fmp12 file on this site.

Once the two custom functions of Developer() and Debug() have been added to your solution, you can easily copy and paste the button to any layout. By setting the Inspector palette Position > Autosizing > Locks to anchor the De"bug" button to the rightmost edge, you can easily move this button within the viewable and non-viewable areas of the layout while developing. You can also leverage the ability to use conditional formatting to hide the image on the button using the Developer() custom function. Although, this is not suggested when a solution is released. It’s best just to move the debug button off the viewable portion of the layout. Here is a screenshot of the initial layout for the Standards.fmp12 file.

The De"bug" button can also use the Debug() function within its tooltip setting. This makes it easy to hover over the bug icon and get valuable information.