FM.Loop function

For years we wrote loops in FileMaker scripts to count up from one value to other value and call a plugin function for each time. But that needs always 5 lines for just a simple loop to e.g. fill a table of values. You can loop already by building a custom function with recursion, but that is complicated and can be difficult to debug. And when our scripts use custom functions, they are not as portable, so for the next plugin version we add a FM.Loop function:

MBS("FM.Loop"; Variable Name; Start Value; End Value; Step Value; Expression)

This function takes the name of variable, can be with zero, one or two $ in the name to make a local, script or global variable. Than you pass a start value (or expression), an end value and the step value. So we start with our variable set to start value. Than we loop while variable <= end value and add step value after each loop. You probably noticed, that this is exactly a FOR loop in other programming tools. For each time, we evaluate the given expression, which can of course reference the variable.

Here an example:

```
MBS("FM.Loop";
/* variable */ "index";
/* start */ 1;
/* end */ 12;
/* step */ 1;
/* formula */
"MBS(\"DynaPDF.Table.SetCellText\"; $table; $rowNum; index;
\"center\"; \"center\"; GetValue($months; index))")
```

This calls DynaPDF.Table.SetCellText function with values from 1 to 12 and passes the value from $months list. As you may notice, we need to escape quotes in expression here with \" to have FileMaker not get confused.

```
MBS("FM.Loop";
/* variable */ "index";
/* start */ 1;
/* end */ 31;
/* step */ 1;
```

```
/* formula */
"Let([$rowNum = MBS(\"DynaPDF.Table.AddRow\"; $table);
r = MBS(\"DynaPDF.Table.SetCellText\"; $table; $rowNum; 0;
\"center\"; \"center\"; index)]; r)")
```

Second example uses a Let Statement to run two MBS functions. First line defines variable $rowNum with line number and second call to fill first cell with number of the line. This way we create 31 lines with each having the line number in first cell.

```
MBS("FM.Loop";
/* variable */ "index";
/* start */ 10;
/* end */ 0;
/* step */ -1;
/* formula */
"index/10")
```

In third example we use the fact, that return value is the list of results of expressions, so this function just returns a list with numbers from 10 down 0.
If the expression causes an error, we return the error and cancel early.

Will come next week in 8.2 plugin pre-release download.