Dynamically build MBS call and evaluate it

Here is an example where we dynamically build the command to evaluate with parameters in $P[] indexed variables. The example inserts a copy of the current record into the same table using FM.InsertRecord. This could be done more efficiently with FM.InsertRecordQuery in one line, but here we wantt o show explicitly how to dynamically build parameter list at runtime:

```
# Our input parameters with field name list
Set Variable [ $FieldNames ; Value: FieldNames ( Get(FileName) ;
Get(LayoutName) ) ]
#
# We build parameters for our Evaluate call
Set Variable [ $Params ; Value: "" ]
Set Variable [ $ParamIndex ; Value: 0 ]
#
Set Variable [ $ParamIndex ; Value: $ParamIndex + 1 ]
Set Variable [ $P[$ParamIndex] ; Value: Get(FileName) ]
Set Variable [ $Params ; Value: $Params & "; $P[" & $ParamIndex & "]" ]
#
Set Variable [ $ParamIndex ; Value: $ParamIndex + 1 ]
Set Variable [ $P[$ParamIndex] ; Value: Get(LayoutTableName) ]
Set Variable [ $Params ; Value: $Params & "; $P[" & $ParamIndex & "]" ]
#
# We loop over feidl list
Set Variable [ $Count ; Value: ValueCount ( $FieldNames ) ]
Set Variable [ $FieldIndex ; Value: 0 ]
Set Variable [ $Types ; Value: "" ]
#
Loop
    # get field name and value
    Set Variable [ $FieldIndex ; Value: $FieldIndex + 1 ]
    Set Variable [ $Fieldname ; Value: GetValue($FieldNames; $FieldIndex) ]
    Set Variable [ $Value ; Value: GetField ( $Fieldname) ]
    #
    # Check typ to filter fields we don't want
    Set Variable [ $Typ ; Value: FieldType ( Get(FileName) ; $Fieldname ) ]
    If [ Position ( $Typ; "Global"; 1; 1 ) ≥ 1 or Position ( $Typ; "Summary"; 1; 1 )
≥ 1 or Position($Typ; "storedCalc"; 1; 1) ≥ 1 or $Fieldname = "ID" ]
        # ignore global, statistic and unsaved formula field
        // Show Custom Dialog [ "Typ" ; $FieldName & ": " & $typ ]
    Else
        # We store in $P[] our parameters for evaluate and in $Params the
parameters for Evaluate
```

```
                Set Variable [ $Types ; Value: $Types & ¶ & $FieldName & ": " & $Typ
]
                Set Variable [ $ParamIndex ; Value: $ParamIndex + 1 ]
                Set Variable [ $P[$ParamIndex] ; Value: $Fieldname ]
                Set Variable [ $Params ; Value: $Params & "; $P[" & $ParamIndex &
"]" ]
                Set Variable [ $ParamIndex ; Value: $ParamIndex + 1 ]
                Set Variable [ $P[$ParamIndex] ; Value: $Value ]
                Set Variable [ $Params ; Value: $Params & "; $P[" & $ParamIndex &
"]" ]
        End If
        #
        Exit Loop If [ $FieldIndex = $count ]
End Loop
#
# now build Evaluate command and show it
// Show Custom Dialog [ "Types" ; $Types ]
Set Variable [ $Function ; Value: "FM.InsertRecord" ]
Set Variable [ $command ; Value: "MBS($Function" & $params & ")" ]
Show Custom Dialog [ "Command" ; $Command ]
If [ Get ( LastMessageChoice ) = 1 ]
        # And run it!
        Set Variable [ $result ; Value: Evaluate($Command) ]
        Show Custom Dialog [ "Result" ; $Result ]

End If
```

By passing command with $ variables to evaluate, we avoid converting all parameters to text and our values retain their data type.