Read any SAP Table with Microsoft Excel

Applies to:

The code sample is created on the SAP Web Application Server 6.40 but is compatible with 6.20 and next releases.

Summary

It's a fine example to understand how the macro Excel can call Function RFC in a very simple way.

With a little stress the call could be made to a dynamic function in order to read the content at any table defined in DDIC like a transaction SE16 (reduced version).

This utility could be useful to check in easy way different tables into Excel at the same time from differents system.

So here we have two functions to define in SAP, one to read the data dynamically from any table and the other one to prepare in an include the declaration of the table.

In Excel two macros are defined in order to call the SAP function and render the data on the sheet.

Author: Sergio Locatelli

Company: Techedge

Created on: 06 November 2006

Author Bio

I'm a mathematician. I'm working with SAP since '97, now I'm intresting with SOA concepts and realization. About NW components I worked in particular with BI and XI.

Table of Contents

Function in SAP

There are two necessary functions in SAP system plus one include.

Function 1 - set the table declaration

To retrive the data dynamically with the main function, is necessary to set the declaration of the table in a separated include.

This function redefine the include involved with the table in object.

```
FUNCTION Z_BC_TAB_TABLE_DEC.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(TABLENAME) LIKE  DD03L-TABNAME
*"  EXCEPTIONS
*"      TABLE_NOT_EXIST
*"----------------------------------------------------------------------

  select single tabname into tablename
                from dd02l
                where tabname eq tablename.
  if sy-subrc ne 0.
    raise TABLE_NOT_EXIST.
  endif.

  DATA programm(72) OCCURS 0 WITH HEADER LINE.
  DATA lungh(6).

  FIELD-SYMBOLS <tabella>.

  READ REPORT 'ZBCTAB_TABLE_DECLARE' INTO programm.

  LOOP AT programm.
    CHECK programm(1) NE '*'.
    CONDENSE programm.

    IF programm(7) = 'TABLES '.
      CLEAR programm.
    CONCATENATE 'TABLES' tablename '.' INTO programm SEPARATED BY space.
      MODIFY programm.
    ENDIF.
  ENDLOOP.
  INSERT REPORT 'ZBCTAB_TABLE_DECLARE' FROM programm.



ENDFUNCTION.
```

Include – Table declaration

This include will be redefined at every call of the first function, here there is the start code.

```
*&---------------------------------------------------------------------*
*& Report  ZBCTAB_TABLE_DECLARE
```

```
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*


TABLES T000.
```

Function 2 - extract all data from table

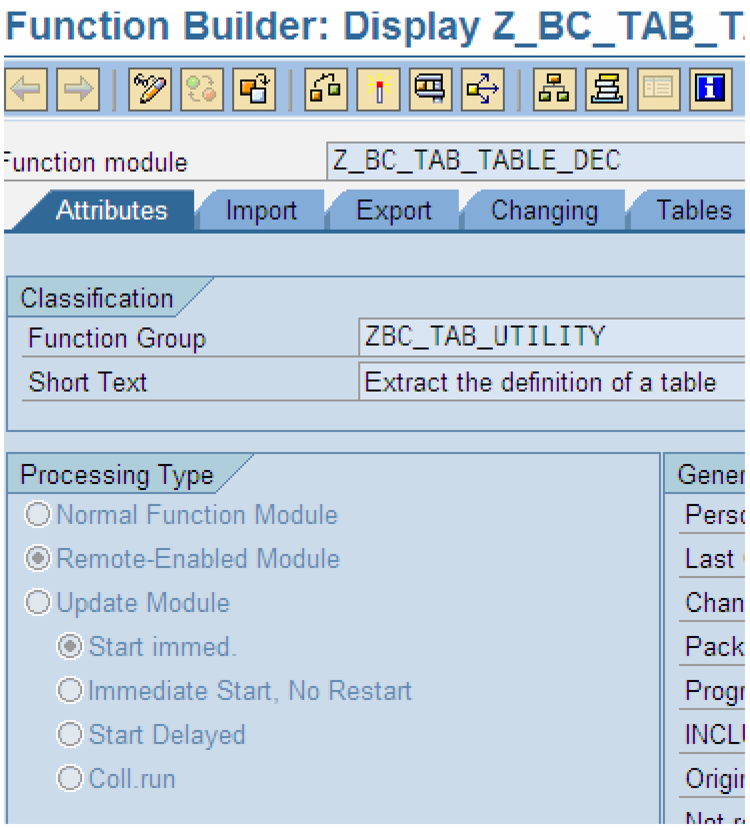
This function extracts the data and the structure of the table from the dictionary.

Prerequisite is that the table to query is declared in the include shared by this function.

```
FUNCTION Z_BC_TAB_TABLE.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(N_RECORD) LIKE  RSEUMOD-TBMAXSEL DEFAULT 200
*"     VALUE(TABLENAME) LIKE  DD03L-TABNAME
*"     VALUE(N_FIELD) LIKE  RSEUMOD-TBMAXSEL DEFAULT 10
*"     VALUE(CONDITION) LIKE  TCUSDAT-VALUE
*"  TABLES
*"      TABLECONTENT STRUCTURE  ZCHART1250
*"      TABLESTRUCT STRUCTURE  DD03L
*"      TABLESTEXT STRUCTURE  DD03T OPTIONAL
*"  EXCEPTIONS
*"      TABLE_NOT_DECLARED
*"      NO_RECORD_FOUND
*"----------------------------------------------------------------------
* Note: only declared table in TOP could be query dinamically

  include ZBCTAB_TABLE_DECLARE.
  DATA: count LIKE n_field.
  data: offset type i.
  data: index like sy-index.

  DATA: BEGIN OF tablestruct_i.
          INCLUDE STRUCTURE dd03l.
  DATA: END OF tablestruct_i.
  DATA: BEGIN OF tablestruct_t.
          INCLUDE STRUCTURE dd03t.
  DATA: END OF tablestruct_t.


  DATA:
      ftab TYPE TABLE OF string.
  DATA: fieldnam LIKE dd03l-fieldname.

  FIELD-SYMBOLS <f> TYPE ANY.
  FIELD-SYMBOLS <f2> TYPE ANY.
  DATA: content_t LIKE zchart1250.

  SELECT * FROM dd03l  INTO tablestruct_i
                WHERE tabname = tablename
                ORDER BY position.
    check tablestruct_i-fieldname ns 'INCLUDE'.
    ADD 1 TO count.
    APPEND tablestruct_i TO tablestruct.
    APPEND tablestruct_i-fieldname TO ftab.

    select single * from dd03t into tablestruct_t
                WHERE tabname = tablename
                and DDLANGUAGE = sy-langu
                and FIELDNAME = tablestruct_i-FIELDNAME.
    if sy-subrc ne 0.
    select single DDTEXT into tablestruct_t-DDTEXT
           from dd04t
                WHERE ROLLNAME = tablestruct_i-ROLLNAME
                and DDLANGUAGE = sy-langu.
*                and FIELDNAME = tablestruct_i-FIELDNAME.

    endif.
    APPEND tablestruct_t TO TABLESTEXT.

    IF count GE n_field.
      EXIT.
    ENDIF.


  ENDSELECT.

  ASSIGN TABLE FIELD (tablename) TO <f>.
  if sy-subrc ne 0.
    raise TABLE_NOT_DECLARED.
  endif.
  count = 0.

  SELECT DISTINCT (ftab)
       INTO CORRESPONDING FIELDS OF <f>
       FROM (tablename)
       where (condition).

    clear content_t-string.
    clear offset.
    add 1 to count.
    loop at tablestruct into tablestruct_i.
     assign component sy-tabix of structure <f> to <F2>.
     write <F2> to content_t-string+offset.
*      concatenate content_t-string <F2> into content_t-string.
     offset = offset + tablestruct_i-leng.
    endloop.
    APPEND content_t TO tablecontent.
    if count ge N_RECORD.
      exit.
    endif.
  ENDSELECT.
```

```
ENDFUNCTION.
```

Be sure that your function has the option Remote-Enabled Module.


Function Builder: Display Z_BC_TAB_T

Function module Z_BC_TAB_TABLE_DEC

Classification

Function Group	ZBC_TAB_UTILITY
Short Text	Extract the definition of a table

Processing Type

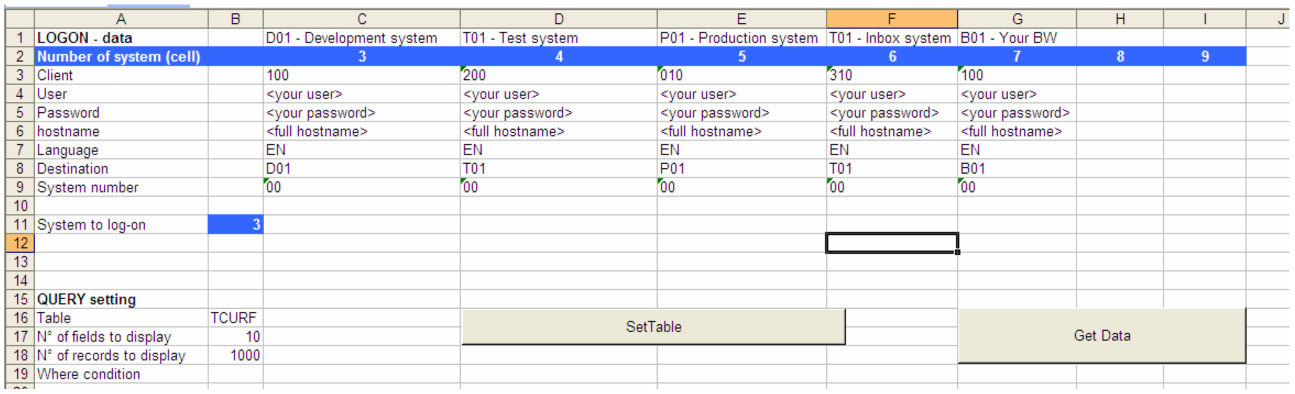
( ) Normal Function Module
(●) Remote-Enabled Module
( ) Update Module
(●) Start immed.
( ) Immediate Start, No Restart
( ) Start Delayed
( ) Coll.run

Excel file

Sheet Query setting

The first sheet in the file will contains the info of logon for different system and the selection data for

It must be like the following:



	A	B	C	D	E	F	G	H	I	J
1	**LOGON - data**		D01 - Development system	T01 - Test system	P01 - Production system	T01 - Inbox system	B01 - Your BW			
2	**Number of system (cell)**		**3**	**4**	**5**	**6**	**7**	**8**	**9**	
3	Client		100	200	010	310	100			
4	User		<your user>	<your user>	<your user>	<your user>	<your user>			
5	Password		<your password>	<your password>	<your password>	<your password>	<your password>			
6	hostname		<full hostname>	<full hostname>	<full hostname>	<full hostname>	<full hostname>			
7	Language		EN	EN	EN	EN	EN			
8	Destination		D01	T01	P01	T01	B01			
9	System number		00	00	00	00	00			
10										
11	System to log-on	3								
12										
13										
14										
15	**QUERY setting**									
16	Table	TCURF								
17	N° of fields to display	10								
18	N° of records to display	1000								
19	Where condition									

Note that in the macros the position of the cells are fixed, so if you intend to shift or change the disposition of the cells you must change also the macros.

Explanation:

Here you can maintain the logon data for different systems, as in the example you can have a development syste, a quality system, a production, an inbox, and so on.

On the rows Client, hostaname, language, destination and system number there are the info that you have in your SapGui

In the rows User and password you can write your personal user/password but pay attention these are sensible data here and can be hidden in different way. I don't go into details here.

In the row 2 I numbered the system to simple identification.

In the row 11 at "system to log-on" insert the identification of the system you would log-on.

Below in rows 16-19 there are the selection data for the specific query:

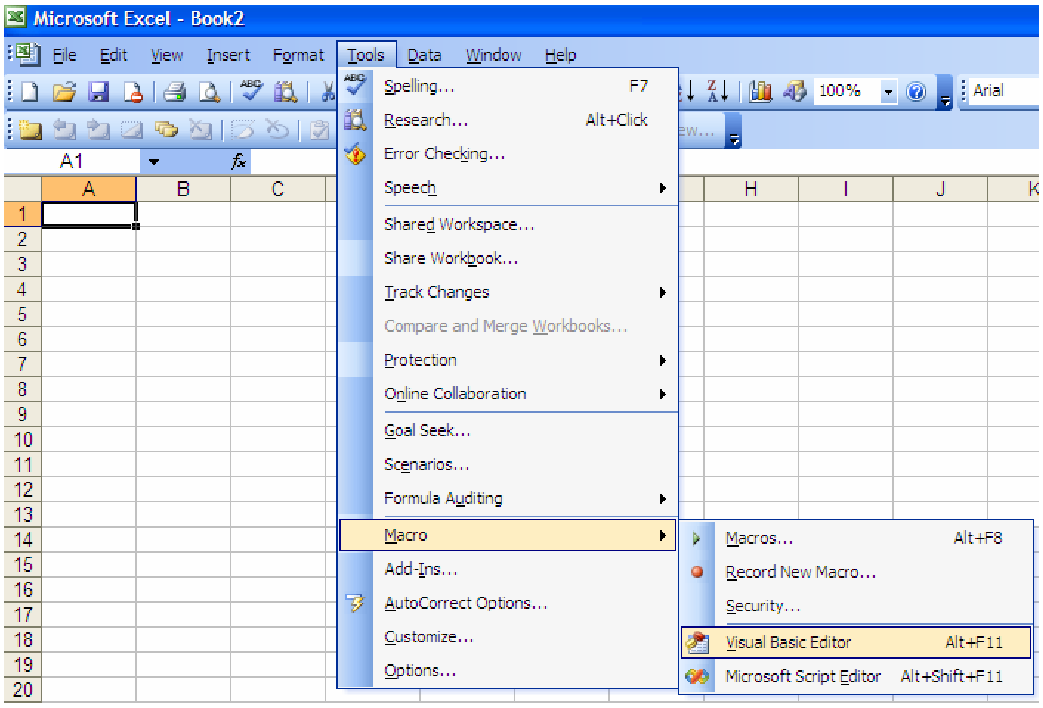
- Table
- N° of fields to display
- N° of records to display (cause of performance and Excel-limitation)
- Where condition = is a free cell in which you can define a simple where condition to filter the data

Sheet Query Result

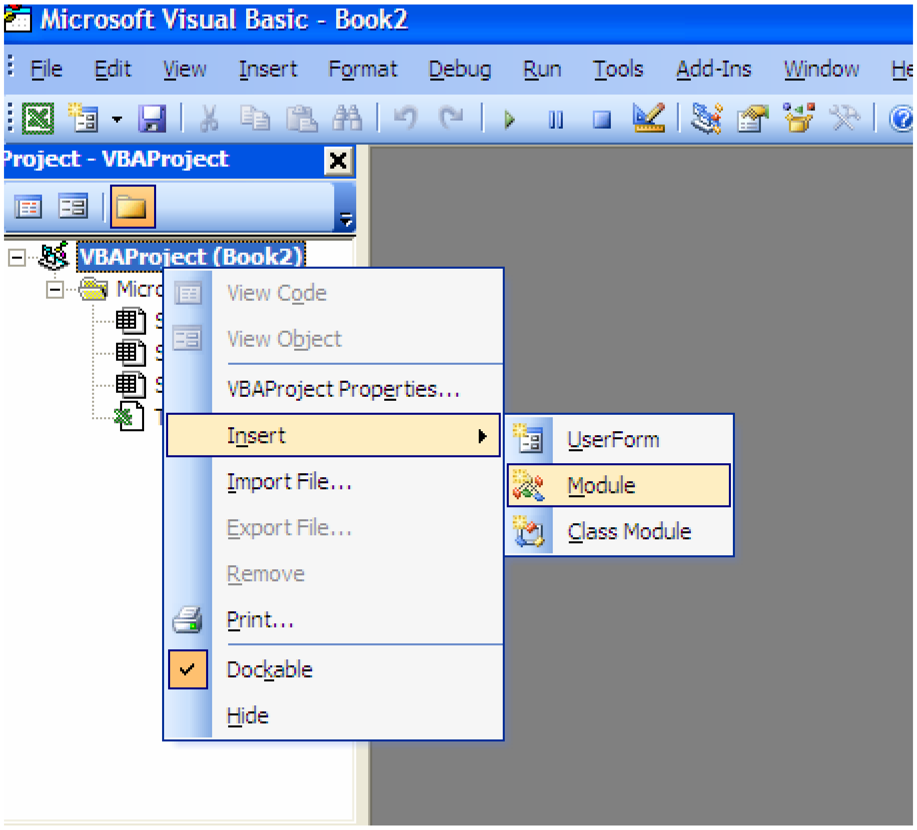
That's the second sheet in the Excel file. It will be filled directly from the macros after a successful query from SAP.

VBA Macros in Excel

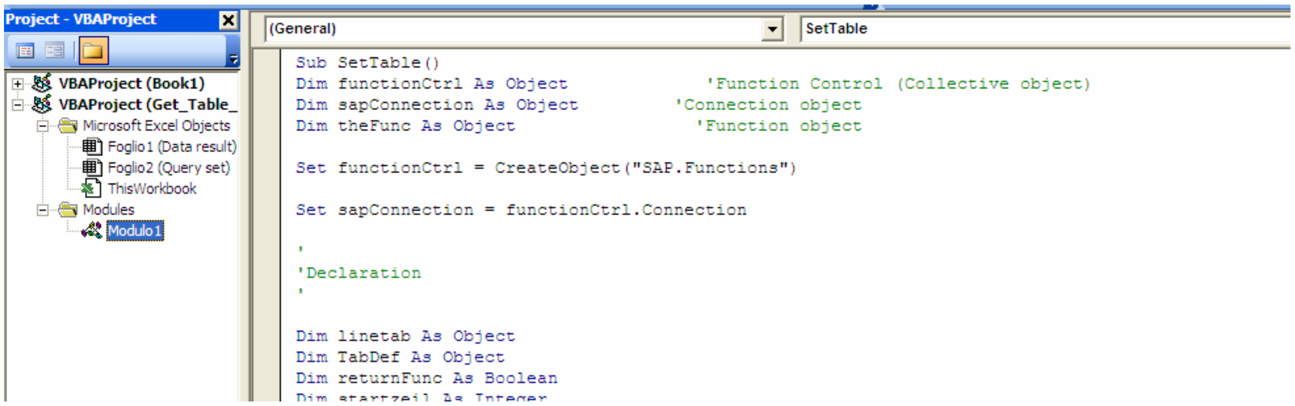
In the Excel file, go to VBA editor:



insert a new module:



In the editor insert the code reported in the article



```
Sub SetTable()
Dim functionCtrl As Object            'Function Control (Collective object)
Dim sapConnection As Object        'Connection object
Dim theFunc As Object                'Function object

Set functionCtrl = CreateObject("SAP.Functions")

Set sapConnection = functionCtrl.Connection

'
'Declaration
'

Dim linetab As Object
Dim TabDef As Object
Dim returnFunc As Boolean
Dim startzeil As Integer
```

****

```
Sub SetTable()
Dim functionCtrl As Object            'Function Control (Collective object)
Dim sapConnection As Object        'Connection object
Dim theFunc As Object                'Function object

Set functionCtrl = CreateObject("SAP.Functions")

Set sapConnection = functionCtrl.Connection

'
'Declaration
'

Dim linetab As Object
Dim TabDef As Object
Dim returnFunc As Boolean
Dim startzeil As Integer
Dim endcol As Integer
Dim table_name As String
Dim n_record As String
Dim n_fields As String
Dim where_condition As String
Dim start_char As Integer
Dim WriteCell As String
Dim Offset As Integer
Dim Destination_System As Integer


'
'Logon with initial values
'
Destination_System = ActiveSheet.Cells(11, 2).Value

sapConnection.client = ActiveSheet.Cells(3, Destination_System).Value
sapConnection.user = ActiveSheet.Cells(4, Destination_System).Value
sapConnection.Language = ActiveSheet.Cells(7, Destination_System).Value
sapConnection.hostname = ActiveSheet.Cells(6, Destination_System).Value
sapConnection.Password = ActiveSheet.Cells(5, Destination_System).Value
sapConnection.SystemNumber = ActiveSheet.Cells(9, Destination_System).Value
sapConnection.System = ActiveSheet.Cells(8, Destination_System).Value
sapConnection.Destination = ActiveSheet.Cells(8, Destination_System).Value


If sapConnection.logon(0, False) <> True Then
   MsgBox "No connection to R/3!"
   Exit Sub                                  'End program
End If

  Set theFunc = functionCtrl.Add("Z_BC_TAB_TABLE_DEC")


  table_name = ActiveSheet.Cells(16, 2).Value

  theFunc.exports("TABLENAME") = table_name
  returnFunc = theFunc.Call

  die_exception = theFunc.Exception




End Sub
Sub GetTableContent()

Dim functionCtrl As Object            'Function Control (Collective object)
Dim sapConnection As Object           'Connection object
Dim theFunc As Object                 'Function object

Set functionCtrl = CreateObject("SAP.Functions")

Set sapConnection = functionCtrl.Connection

'
'Declaration
'

Dim linetab As Object
Dim TabDef As Object
Dim TabDefName As Object
Dim returnFunc As Boolean
Dim startzeil As Integer
Dim endcol As Integer
Dim table_name As String
Dim n_record As String
Dim n_fields As String
Dim where_condition As String
Dim start_char As Integer
Dim WriteCell As String
Dim Offset As Integer

'
'Logon with initial values
'
Destination_System = ActiveSheet.Cells(11, 2).Value
```

```
sapConnection.client = ActiveSheet.Cells(3, Destination_System).Value
sapConnection.user = ActiveSheet.Cells(4, Destination_System).Value
sapConnection.Language = ActiveSheet.Cells(7, Destination_System).Value
sapConnection.hostname = ActiveSheet.Cells(6, Destination_System).Value
sapConnection.Password = ActiveSheet.Cells(5, Destination_System).Value
sapConnection.SystemNumber = ActiveSheet.Cells(9, Destination_System).Value
sapConnection.System = ActiveSheet.Cells(8, Destination_System).Value
sapConnection.Destination = ActiveSheet.Cells(8, Destination_System).Value


If sapConnection.logon(0, False) <> True Then
   MsgBox "No connection to R/3!"
   Exit Sub                            'End program
End If


  Set theFunc = functionCtrl.Add("Z_BC_TAB_TABLE")

  n_fields = ActiveSheet.Cells(17, 2).Value
  n_record = ActiveSheet.Cells(18, 2).Value
  where_condition = ActiveSheet.Cells(19, 2).Value
  table_name = ActiveSheet.Cells(16, 2).Value


'Prepare output to the EXCEL worksheet
'
Worksheets(2).Select
Cells.Clear



startzeil = 1


'Determine the import parameters for the function call
'
' For start_char = Asc("A") To Asc("Z")

        theFunc.exports("TABLENAME") = table_name
        theFunc.exports("N_FIELD") = n_fields
        theFunc.exports("N_RECORD") = n_record
        theFunc.exports("CONDITION") = where_condition

        returnFunc = theFunc.Call

        die_exception = theFunc.Exception

         If returnFunc = True Then
            Set linetab = theFunc.Tables.Item("TABLECONTENT")
            Set TabDef = theFunc.Tables.Item("TABLESTRUCT")
            Set TabDefName = theFunc.Tables.Item("TABLESTEXT")
            endcol = 0
            Call display_header(TabDef, TabDefName, n_fields)
            Call display_lines(table_name, linetab, TabDef, startzeil, endcol)
            startzeil = endcol
            Set customers = Nothing
        Else
            If die_exception = "NO_RECORD_FOUND" Then
               Cells(startzeil, 1) = "No values exist for " + the_name
                startzeil = startzeil + 1
            Else
                MsgBox "Error when accessing function in R/3 ! "
            Exit Sub
            End If
        End If


'Close connection to R/3 !
'
functionCtrl.Connection.logoff

'
'Release the objects to free storage space
'
Set sapConnection = Nothing
Set functionCtrl = Nothing

MsgBox "Program terminated!", 0, "Exit"


End Sub
Sub display_header(ByRef table_def As Object, ByRef table_name As Object, n_fields As String)
'
'Show table header
'For each field, the name and the description.

  j = 1
  For Each TabDef In table_def.Rows
    Cells(1, j) = Trim(TabDef("FIELDNAME"))
    j = j + 1
  Next
  j = 1
  For Each TabDefName In table_name.Rows
    Cells(2, j) = Trim(TabDefName("DDTEXT"))
    j = j + 1
  Next


End Sub
Sub display_lines(TabName As String, ByRef line_table As Object, ByRef table_def As Object, start_zeil As
Integer, ByRef end_col As Integer)

'
'Display contents of customer table
'

bManyLines = False
If (bManyLines = False) Then
i = 3
For Each Line In line_table.Rows
  Offset = 1
  j = 1
  For Each TabDef In table_def.Rows
    Leng = Trim(TabDef("LENG"))
    WriteCell = Mid(Trim(Line("STRING")), Offset, Leng)
    Cells(i, j) = WriteCell
    Offset = Offset + Leng
    j = j + 1
  Next
i = i + 1
Next
End If

end_col = i

End Sub
```

Description of Module

The routine are quite simple, two are the mains:

Routine: SetTable – Call the first function in SAP

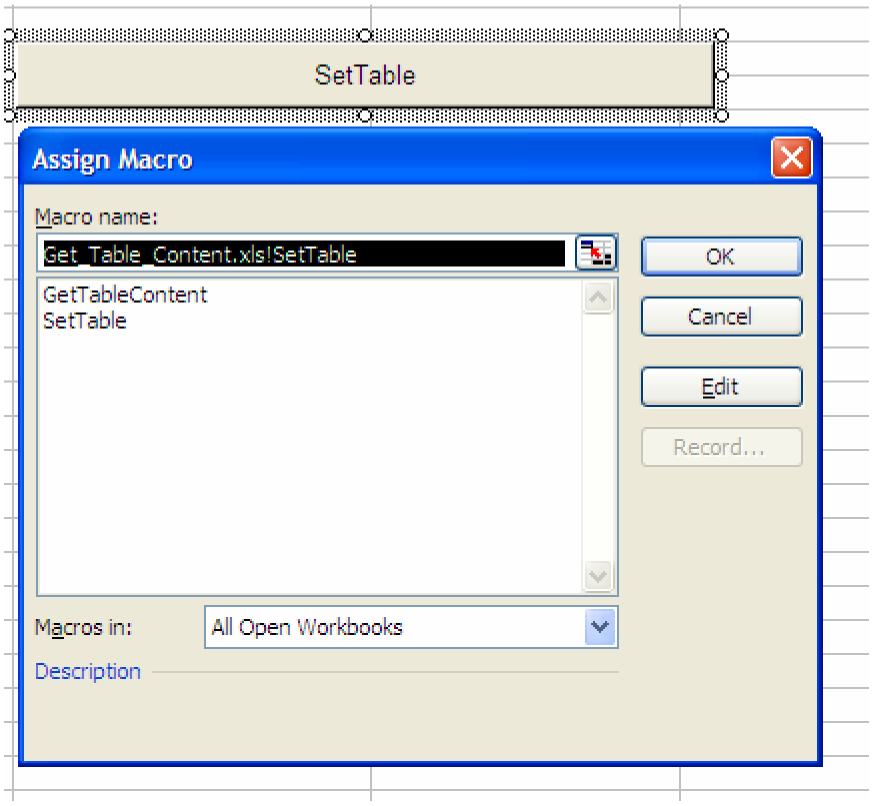
Routine: GetTableContent – Call the second function in SAP to retrive the data and display it

The routine display_header and display_lines are used by GetTableContent to display the result.

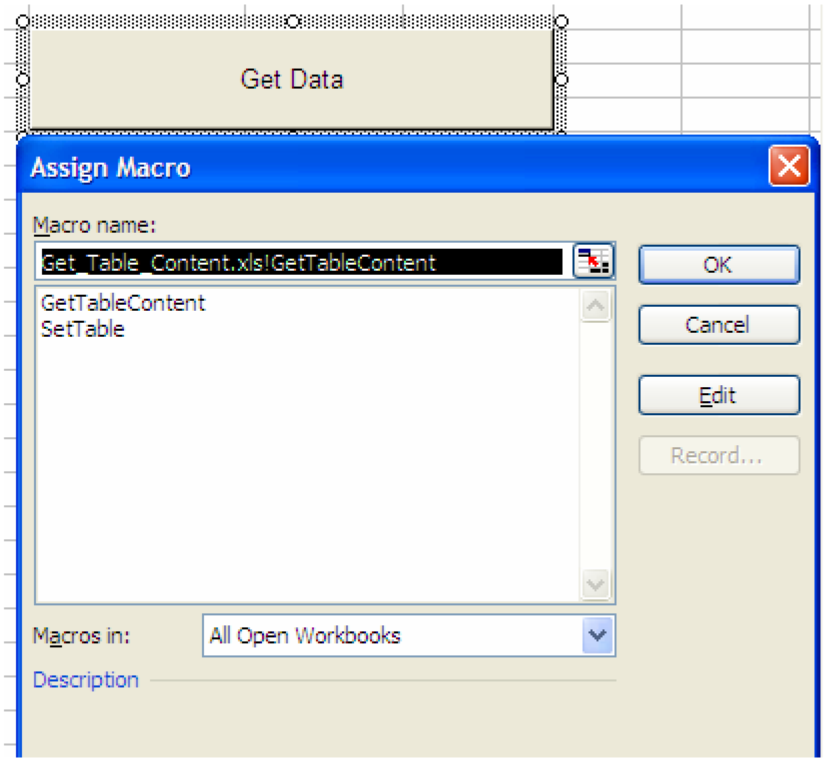
Assign Macro to Button

Assign to the buttons the two main macros:

SetTable



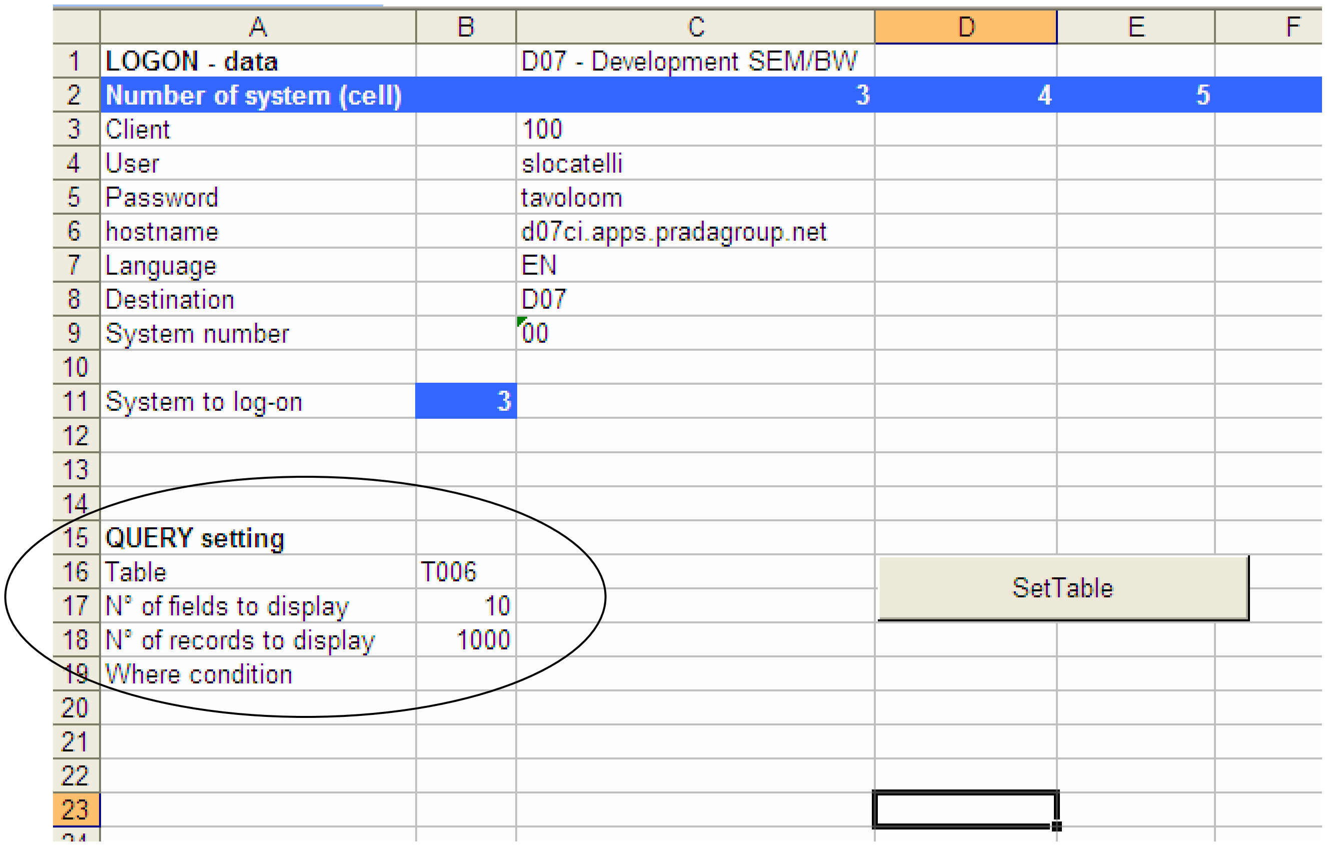
And GetTableContent



Procedure to run the Excel

After the complete definition of functions and macros, after the insertion of the logon data, you can read the data of any table.

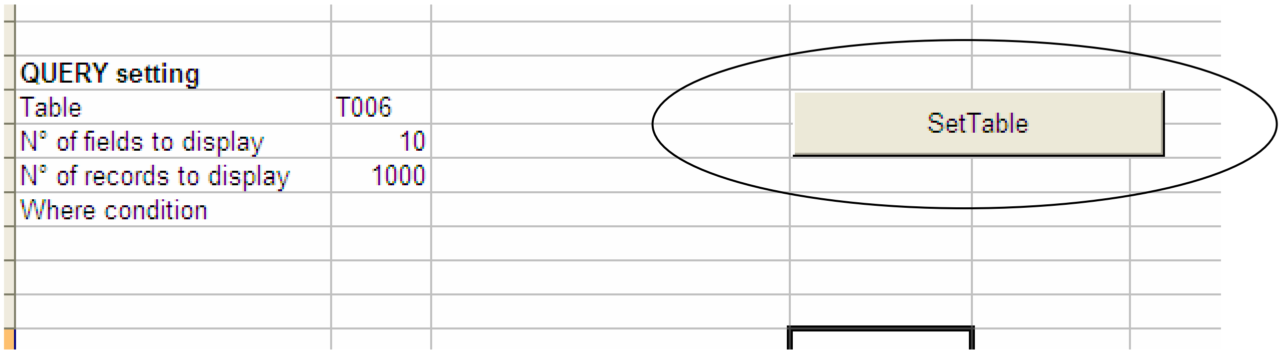
1) Insert the name of the table in query setting



	A	B	C	D	E	F
1	**LOGON - data**		D07 - Development SEM/BW			
2	**Number of system (cell)**		3	4	5	
3	Client		100			
4	User		slocatelli			
5	Password		tavoloom			
6	hostname		d07ci.apps.pradagroup.net			
7	Language		EN			
8	Destination		D07			
9	System number		00			
10						
11	System to log-on	3				
12						
13						
14						
15	**QUERY setting**					
16	Table	T006		SetTable		
17	N° of fields to display	10				
18	N° of records to display	1000				
19	Where condition					
20						
21						
22						
23						

In the where clause you can set an optional condition ABAP to refine the selection (ex. KURST = 'M' AND FCURR = 'EUR')

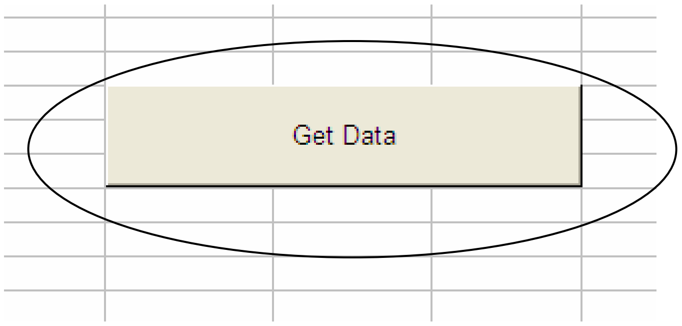
2) Push the button "Set Table"



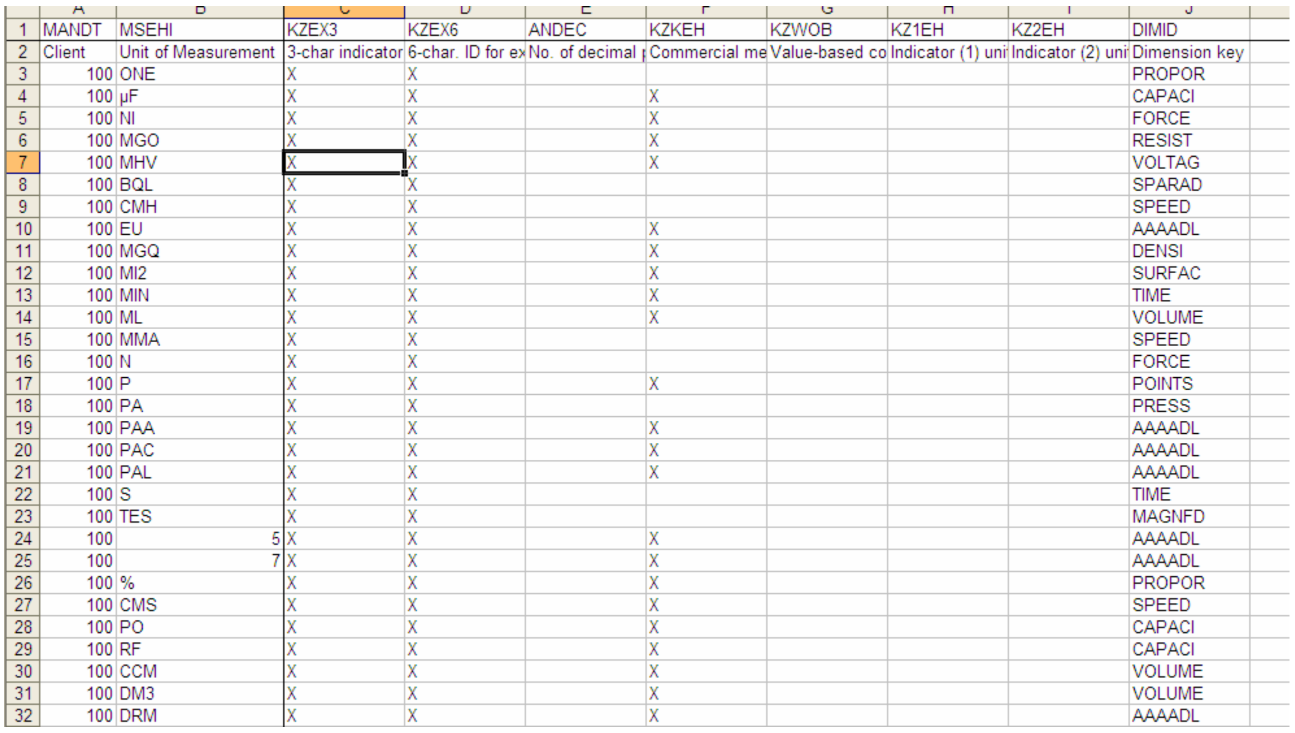
QUERY setting			
Table	T006		SetTable
N° of fields to display	10		
N° of records to display	1000		
Where condition			

This step is necessary separated from the next because to set the table in the include.

3) Push the button " Get data"



The macro sends you in the second sheet with the result of the query.



	A	B	C	D	E	F	G	H	I	J
1	MANDT	MSEHI	KZEX3	KZEX6	ANDEC	KZKEH	KZWOB	KZ1EH	KZ2EH	DIMID
2	Client	Unit of Measurement	3-char indicator	6-char. ID for ex	No. of decimal p	Commercial me	Value-based co	Indicator (1) uni	Indicator (2) uni	Dimension key
3	100	ONE	X	X						PROPOR
4	100	µF	X	X		X				CAPACI
5	100	NI	X	X		X				FORCE
6	100	MGO	X	X		X				RESIST
7	100	MHV	X	X		X				VOLTAG
8	100	BQL	X	X						SPARAD
9	100	CMH	X	X						SPEED
10	100	EU	X	X		X				AAAADL
11	100	MGQ	X	X		X				DENSI
12	100	MI2	X	X		X				SURFAC
13	100	MIN	X	X		X				TIME
14	100	ML	X	X		X				VOLUME
15	100	MMA	X	X						SPEED
16	100	N	X	X						FORCE
17	100	P	X	X		X				POINTS
18	100	PA	X	X						PRESS
19	100	PAA	X	X		X				AAAADL
20	100	PAC	X	X		X				AAAADL
21	100	PAL	X	X		X				AAAADL
22	100	S	X	X						TIME
23	100	TES	X	X						MAGNFD
24	100	5	X	X		X				AAAADL
25	100	7	X	X		X				AAAADL
26	100	%	X	X		X				PROPOR
27	100	CMS	X	X		X				SPEED
28	100	PO	X	X		X				CAPACI
29	100	RF	X	X		X				CAPACI
30	100	CCM	X	X		X				VOLUME
31	100	DM3	X	X		X				VOLUME
32	100	DRM	X	X		X				AAAADL

Limitations

The limits of this sample tool are obvious but is correct to report them here to prevent an incorrect use of the program.

Type of table --> No cluster table could be read with this simple tool

Type of data --> The data type of time are not rendered in the correct way. The amount are displayed in the internal format of SAP. More develop are required in VBA to display correctly this info.

Performance --> The limitiations of Excel are on the maximum nuber of records. This tool is intended as an example and a nice way to retrive and check simple data. No massive extraction is inended.

Other Ideas

Here I suggest some exercises to do starting on this sample code. The results could be useful for business and technical developing.

- Make many sheets like as the source systems. Every sheet will corresponds to one system, press button and import data from all system together. Add one sheet at the end with formulas to compare the data from the different systems but for the same table.

- Assign also the table of text (where it exists) and read also that together the main table. Insert also the language key as selection and render the data in one time.

Disclaimer and Liability Notice