Reading Excel file from Application Server into ABAP Internal Table

Applies to

SAP ABAP Developers who are working on interface project. Developers who has required to read excel data into internal table format for database update.

Summary

This document gives a step by step process to implement code to download excel data into internal table and update the database table.

Table of Contents

1. Applicable scenario

This solution can be implementing in such scenario where we are getting data from third party system in form of Excel documents (.xls or .xlsx) format. If we have requirement to download data into ABAP internal table for manipulation and updating the database table.

2. Prerequisites

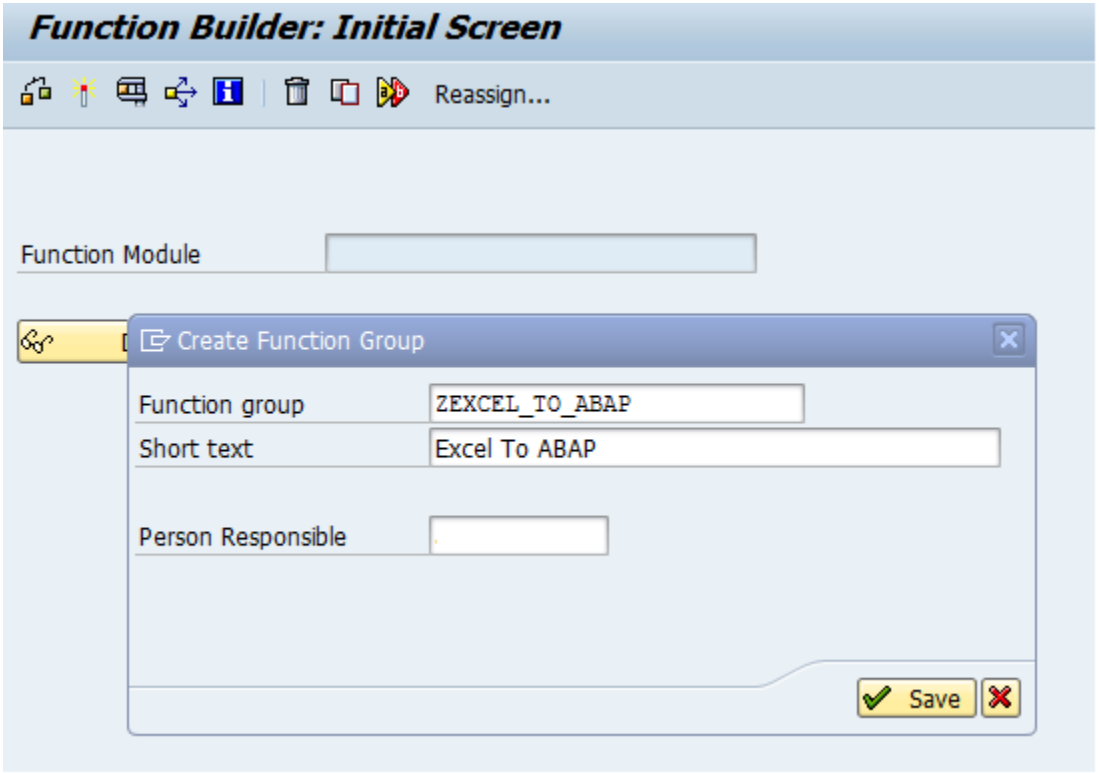
Basic knowledge of ABAP and developer should be able to understand object oriented ABAP code and implement the solution.

3. Introduction

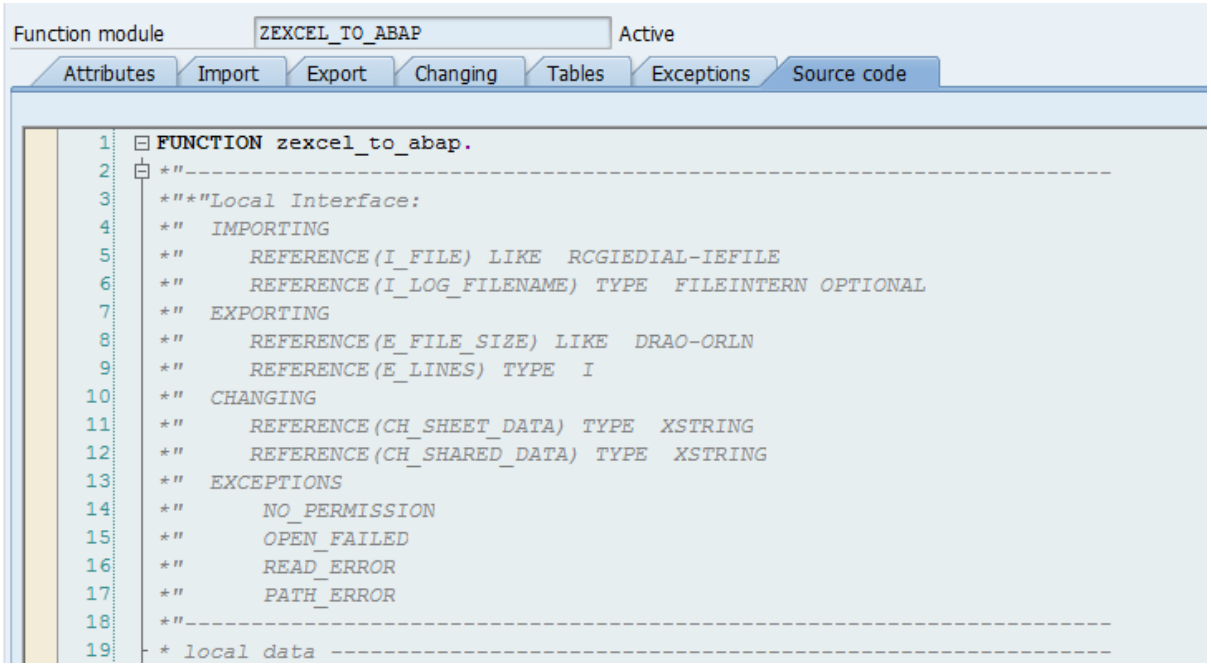
In this document, we will share knowledge on how to extract data from application server on directory path in case we have to download excel file written on application server.

4. Creation of Function Module

In SE37, create the function group and function module.



Please fill these details in Import, Export, Changing and Exceptions parameters.



5. Implementing ABAP code in Program

Source Code –

Global Data –

```
*Constants
CONSTANTS: c_logical_filename_ftappl_2 LIKE filename-fileintern
                                       VALUE 'EHS_FTAPPL_2',
           c_codepage                   TYPE abap_encod VALUE '1160',
           c_sheet_xml                  TYPE i VALUE 2,
           c_shared_str_xml             TYPE i VALUE 3,
           c_filefmt_binary             TYPE rlgrap-filetype VALUE 'BIN'.

*Global data
DATA:g_ex_root TYPE REF TO cx_root,
     g_msg     TYPE string.
```

Function module Source Code -

```
DATA: l_log_filename TYPE fileintern,
       l_stack_tab    TYPE sys_callst,
       l_stack_wa     TYPE sys_calls.
```

```
  DATA : l_len TYPE sy-tabix.
  DATA : l_filename TYPE  authb-filename.
  DATA l_subrc TYPE sy-subrc.

  DATA: l_lines       TYPE i.
  DATA: l_xstring  TYPE xstring.
  DATA  lo_package    TYPE REF TO cl_openxml_package.
  DATA  lo_parts      TYPE REF TO cl_openxml_partcollection.
  DATA  l_sheet_data TYPE xstring.
  DATA  l_shared_data TYPE xstring.

  DATA: li_e_rcgrepfile_tab TYPE cpt_x255,
        li_xtab             TYPE cpt_x255.

  DATA lst_rcgrepfile TYPE cps_x255.

  DATA: lo_xml_part     TYPE REF TO cl_openxml_part,
        lo_xml_part_uri TYPE REF TO cl_openxml_parturi,
        lx_root         TYPE REF TO cx_root,
        l_uri           TYPE string.

  DATA: l_file_content TYPE xstring.

* function body ------------------------------------------------------

* assign value
  l_filename = i_file.

* check the authority for file
  CALL FUNCTION 'AUTHORITY_CHECK_DATASET'
    EXPORTING
*     PROGRAM          =
      activity         = sabc_act_read
* Authority Check allows right now only 60 Character
      filename         = l_filename(60)
    EXCEPTIONS
      no_authority     = 1
      activity_unknown = 2
      OTHERS           = 3.
  IF sy-subrc <> 0.
    RAISE no_permission.
  ENDIF.

  l_log_filename = c_logical_filename_ftappl_2.
```

```
  TRY.
* read the raw-file from the appl.server
      CLEAR l_subrc.

      OPEN DATASET i_file FOR INPUT IN BINARY MODE.
      l_subrc = sy-subrc.
      IF sy-subrc <> 0 OR
          l_subrc <> 0.
        RAISE open_failed.
      ENDIF.

      DO.
        CLEAR l_len.
        CLEAR lst_rcgrepfile.
        READ DATASET i_file INTO lst_rcgrepfile LENGTH l_len.
        IF sy-subrc <> 0.
          IF l_len > 0.
            e_file_size = e_file_size + l_len.
            APPEND lst_rcgrepfile TO li_e_rcgrepfile_tab.
          ENDIF.
          EXIT.
        ENDIF.
        CONCATENATE l_file_content lst_rcgrepfile

          INTO l_file_content IN BYTE MODE.
        e_file_size = e_file_size + l_len.
        APPEND lst_rcgrepfile TO li_e_rcgrepfile_tab.
      ENDDO.

      IF sy-subrc > 10.
        RAISE read_error.
      ENDIF.

      DESCRIBE TABLE li_e_rcgrepfile_tab LINES e_lines.

      CLOSE DATASET i_file.

      IF li_e_rcgrepfile_tab[] IS NOT INITIAL.
        li_xtab[] = li_e_rcgrepfile_tab[].
      ENDIF.

*Convert data to xstring
      cl_scp_change_db=>xtab_to_xstr( EXPORTING im_xtab    = li_xtab
                                                im_size    = l_lines
                                      IMPORTING ex_xstring = l_xstring ).
```

```
*Load document
      lo_package = cl_xlsx_document=>load_document( iv_data = l_xstring ).
*Get parts
      lo_parts = lo_package->get_parts( ).

*Load XML data
      l_uri = lo_parts->get_part( 2 )->get_parts( )-
>get_part( c_sheet_xml )->get_uri( )->get_uri( ).
      lo_xml_part_uri = cl_openxml_parturi=>create_from_partname( l_uri ).
      lo_xml_part = lo_package->get_part_by_uri( lo_xml_part_uri ).
      ch_sheet_data = lo_xml_part->get_data( ).

*Load sheet data
      CLEAR l_uri.
      l_uri = lo_parts->get_part( 2 )->get_parts( )-
>get_part( c_shared_str_xml )->get_uri( )->get_uri( ).
      lo_xml_part_uri = cl_openxml_parturi=>create_from_partname( l_uri ).
      lo_xml_part = lo_package->get_part_by_uri( lo_xml_part_uri ).
      ch_shared_data = lo_xml_part->get_data( ).
    CATCH cx_root INTO g_ex_root.
      CLEAR g_msg.
      g_msg = g_ex_root->get_text( ).
      IF g_msg IS NOT INITIAL.
        sy-subrc = 4.
      ENDIF.
  ENDTRY.
```

Transform XML data to internal table with help of XSLT transformation –

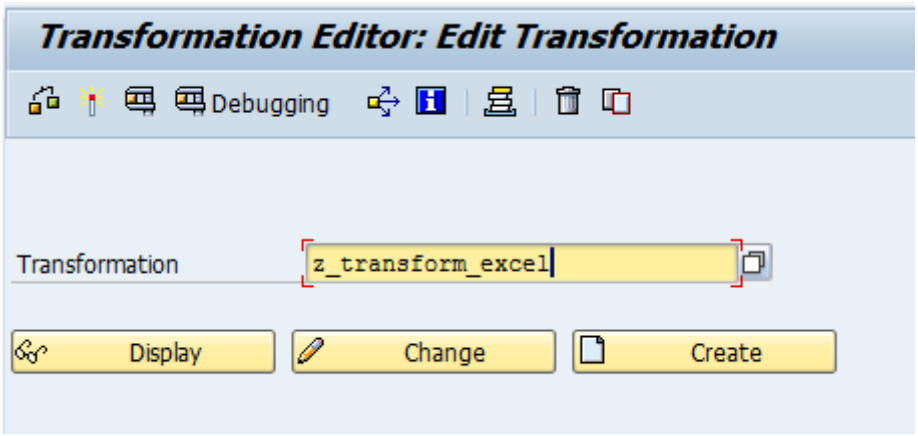
```
*Local data declaration
  DATA lo_shared_str_dom TYPE REF TO if_ixml_document.
  DATA lo_shared_str_nodeset TYPE REF TO if_ixml_node.
  DATA l_shared_str_xml TYPE xstring.
*Converting XML into internal table
  TRY.
      CALL TRANSFORMATION z_transform_excel
        SOURCE XML g_shared_data
        RESULT XML l_shared_str_xml.
*XML to ABAP
      CALL FUNCTION 'SDIXML_XML_TO_DOM'
        EXPORTING
          xml           = l_shared_str_xml
        IMPORTING
          document      = lo_shared_str_dom
        EXCEPTIONS
          invalid_input = 1
          OTHERS        = 2.
      IF sy-subrc = 0.
        lo_shared_str_nodeset = lo_shared_str_dom->clone( ).
      ENDIF.
*Import data
```

```
    CALL TRANSFORMATION z_trans_import_xls
         PARAMETERS
          p_shared_string = lo_shared_str_nodeset
         SOURCE XML g_sheet_data
         RESULT lt_data = i_data1.
  CATCH cx_xslt_exception.
ENDTRY.
```

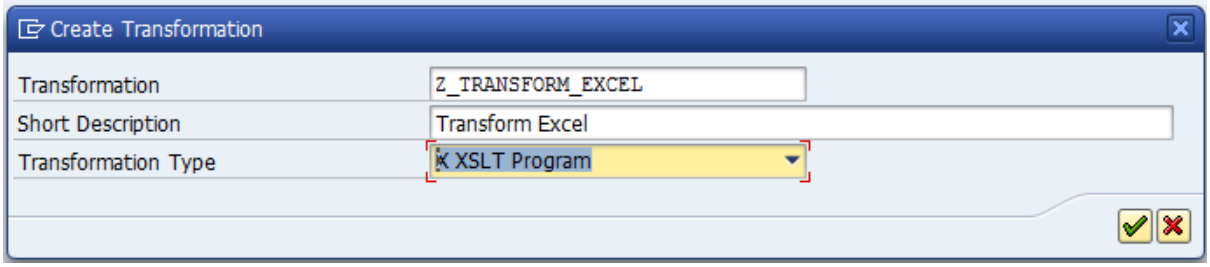
6. Creation of XSLT transformations

Transaction Code – XSLT_TOOL

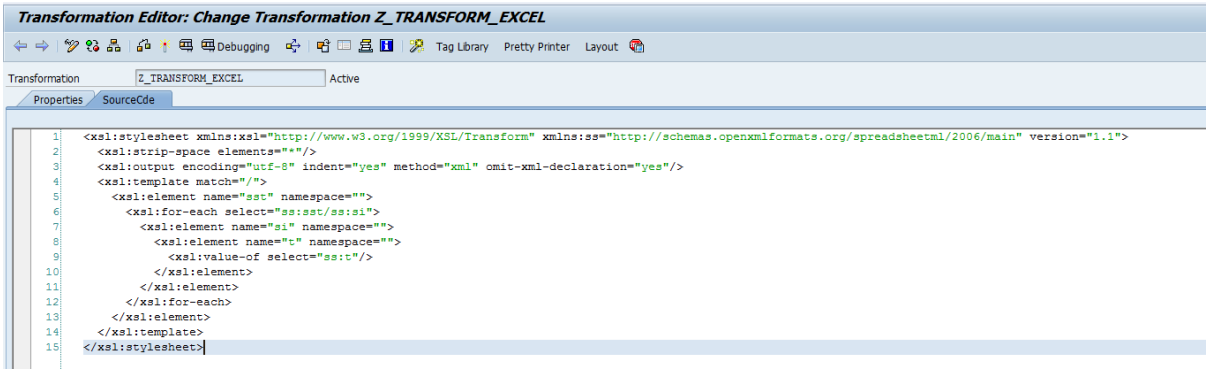
- Step - 1



- Step – 2



- Step - 3

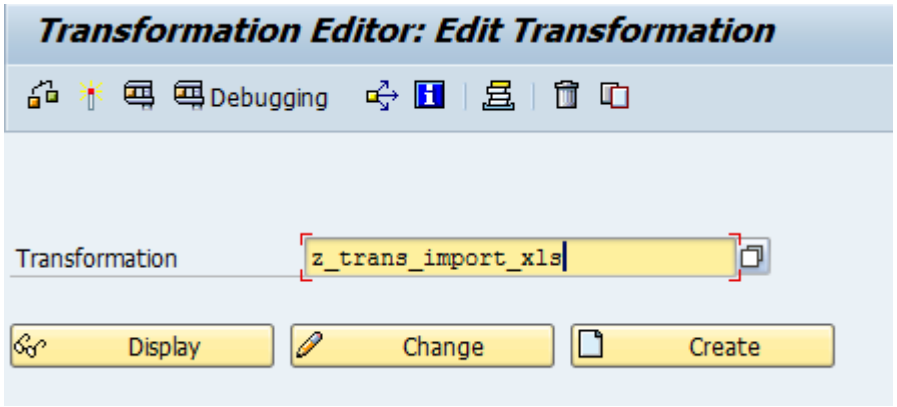


Insert below code and activate the XSLT Transformation -

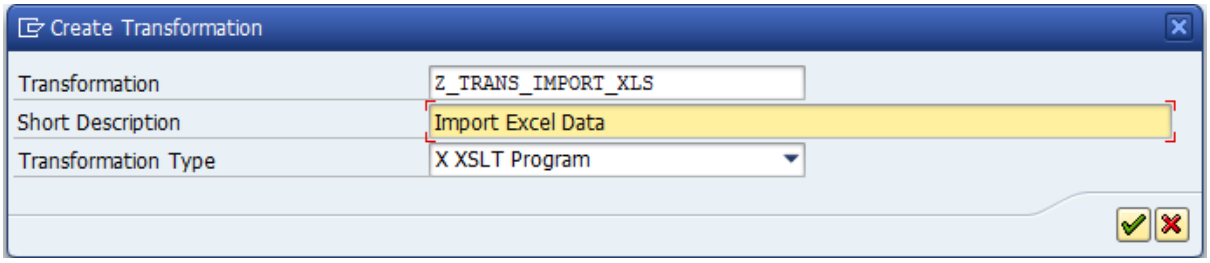
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:ss="
http://schemas.openxmlformats.org/spreadsheetml/2006/main" version="1.1">
  <xsl:strip-space elements="*"/>
  <xsl:output encoding="utf-8" indent="yes" method="xml" omit-xml-
declaration="yes"/>
  <xsl:template match="/">
    <xsl:element name="sst" namespace="">
      <xsl:for-each select="ss:sst/ss:si">
        <xsl:element name="si" namespace="">
          <xsl:element name="t" namespace="">
            <xsl:value-of select="ss:t"/>
          </xsl:element>
        </xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

In similar way, create one more XSLT transformation –

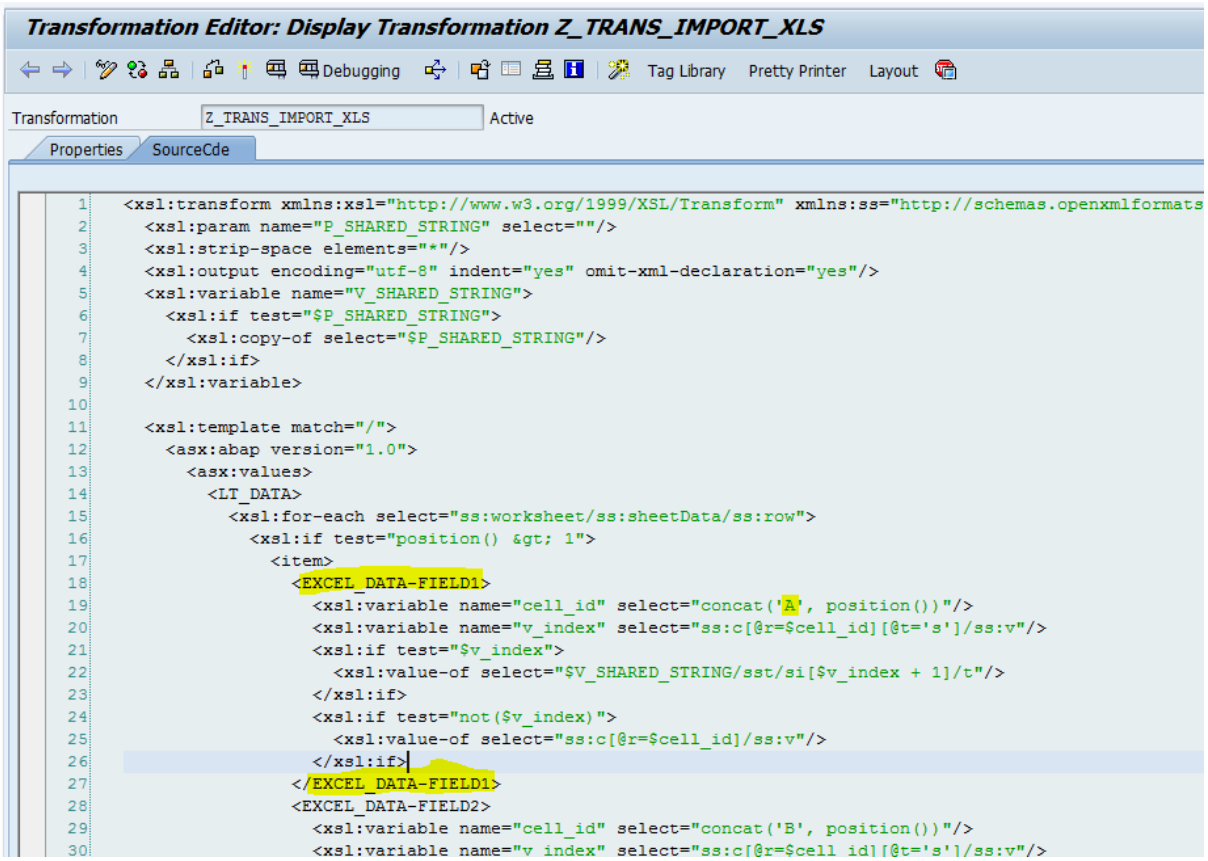
- Step – 1



- Step – 2



- Step - 3



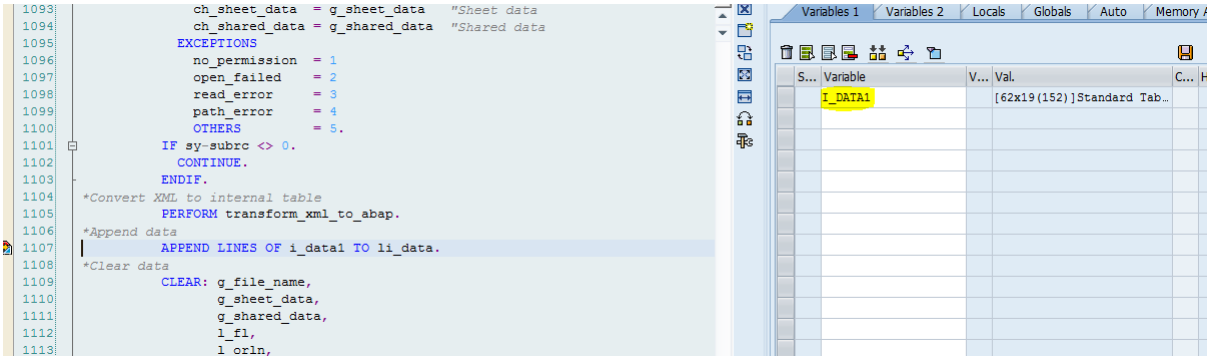

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:ss="http://schemas.openxmlformats
  <xsl:param name="P_SHARED_STRING" select=""/>
  <xsl:strip-space elements="*"/>
  <xsl:output encoding="utf-8" indent="yes" omit-xml-declaration="yes"/>
  <xsl:variable name="V_SHARED_STRING">
    <xsl:if test="$P_SHARED_STRING">
      <xsl:copy-of select="$P_SHARED_STRING"/>
    </xsl:if>
  </xsl:variable>

  <xsl:template match="/">
    <asx:abap version="1.0">
      <asx:values>
        <LT_DATA>
          <xsl:for-each select="ss:worksheet/ss:sheetData/ss:row">
            <xsl:if test="position() &gt; 1">
              <item>
                <EXCEL_DATA-FIELD1>
                  <xsl:variable name="cell_id" select="concat('A', position())"/>
                  <xsl:variable name="v_index" select="ss:c[@r=$cell_id][@t='s']/ss:v"/>
                  <xsl:if test="$v_index">
                    <xsl:value-of select="$V_SHARED_STRING/sst/si[$v_index + 1]/t"/>
                  </xsl:if>
                  <xsl:if test="not($v_index)">
                    <xsl:value-of select="ss:c[@r=$cell_id]/ss:v"/>
                  </xsl:if>
                </EXCEL_DATA-FIELD1>
                <EXCEL_DATA-FIELD2>
                  <xsl:variable name="cell_id" select="concat('B', position())"/>
                  <xsl:variable name="v_index" select="ss:c[@r=$cell_id][@t='s']/ss:v"/>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:ss="h
ttp://schemas.openxmlformats.org/spreadsheetml/2006/main" xmlns:sap="http:/
/www.sap.com/sapxsl" xmlns:asx="http://www.sap.com/abapxml" exclude-result-
prefixes="c" version="1.0">
  <xsl:param name="P_SHARED_STRING" select=""/>
  <xsl:strip-space elements="*"/>
  <xsl:output encoding="utf-8" indent="yes" omit-xml-declaration="yes"/>
  <xsl:variable name="V_SHARED_STRING">
    <xsl:if test="$P_SHARED_STRING">
      <xsl:copy-of select="$P_SHARED_STRING"/>
    </xsl:if>
  </xsl:variable>

  <xsl:template match="/">
    <asx:abap version="1.0">

      <asx:values>
        <LT_DATA>
          <xsl:for-each select="ss:worksheet/ss:sheetData/ss:row">
            <xsl:if test="position() &gt; 1">
              <item>
                <EXCEL_DATA-FIELD1>
```

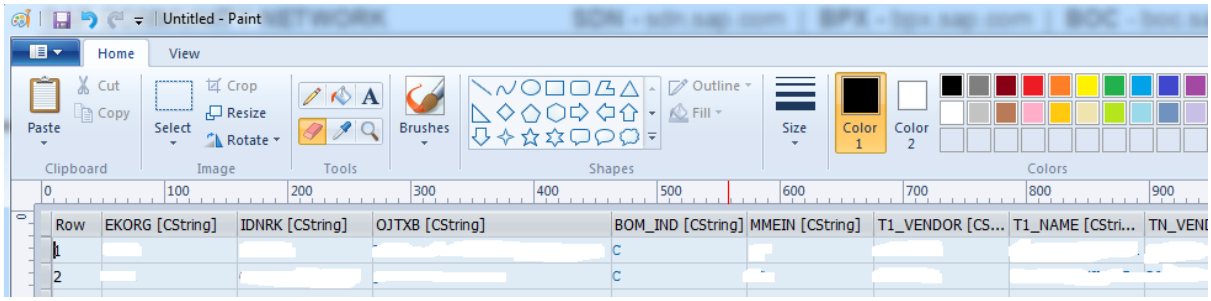
```
                  <xsl:variable name="cell_id" select="concat('A', position
())"/>
                  <xsl:variable name="v_index" select="ss:c[@r=$cell_id][@t
='s']/ss:v"/>
                  <xsl:if test="$v_index">
                    <xsl:value-
of select="$V_SHARED_STRING/sst/si[$v_index + 1]/t"/>
                  </xsl:if>
                  <xsl:if test="not($v_index)">
                    <xsl:value-of select="ss:c[@r=$cell_id]/ss:v"/>
                  </xsl:if>
                </EXCEL_DATA-FIELD1>
                <EXCEL_DATA-FIELD2>
                  <xsl:variable name="cell_id" select="concat('B', position
())"/>
                  <xsl:variable name="v_index" select="ss:c[@r=$cell_id][@t
='s']/ss:v"/>
                  <xsl:if test="$v_index">
                    <xsl:value-
of select="$V_SHARED_STRING/sst/si[$v_index + 1]/t"/>
                  </xsl:if>
                  <xsl:if test="not($v_index)">
                    <xsl:value-of select="ss:c[@r=$cell_id]/ss:v"/>
                  </xsl:if>
                </EXCEL_DATA-FIELD2>
                <EXCEL_DATA-FIELD3>
                  <xsl:variable name="cell_id" select="concat('C', position
())"/>
                  <xsl:variable name="v_index" select="ss:c[@r=$cell_id][@t
='s']/ss:v"/>
                  <xsl:if test="$v_index">
                    <xsl:value-
of select="$V_SHARED_STRING/sst/si[$v_index + 1]/t"/>
                  </xsl:if>
                  <xsl:if test="not($v_index)">
                    <xsl:value-of select="ss:c[@r=$cell_id]/ss:v"/>
                  </xsl:if>
                </EXCEL_DATA-FIELD3>
                <EXCEL_DATA-FIELD4>
                  <xsl:variable name="cell_id" select="concat('D', position
())"/>
                  <xsl:variable name="v_index" select="ss:c[@r=$cell_id][@t
='s']/ss:v"/>
                  <xsl:if test="$v_index">
                    <xsl:value-
```

```
of select="$V_SHARED_STRING/sst/si[$v_index + 1]/t"/>
                  </xsl:if>
                  <xsl:if test="not($v_index)">
                    <xsl:value-of select="ss:c[@r=$cell_id]/ss:v"/>
                  </xsl:if>
                </EXCEL_DATA-FIELD4>
              </item>
            </xsl:if>
          </xsl:for-each>
        </LT_DATA>
      </asx:values>
    </asx:abap>
  </xsl:template>
</xsl:transform>
```

Data appearing in internal table -



Result –


Untitled - Paint
Home
View
Paste
Cut
Copy
Select
Crop
Resize
Rotate
Brushes
Outline
Fill
Size
Color 1
Color 2
Clipboard
Image
Tools
Shapes
Colors
0
100
200
300
400
500
600
700
800
900
Row
EKORG [CString]
IDNRK [CString]
OJTXB [CString]
BOM_IND [CString]
MMEIN [CString]
T1_VENDOR [CS...
T1_NAME [CStri...
1
C
2
C

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.