Mapping an IDOC Schema to an Industry Standard XML Schema

SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

Mapping an IDOC Schema to an Industry Standard XML Schema

Applies To:

SAP R/3 release 4.6C or later.

Summary

Here is the code sample that will render an IDOC schema onto a spreadsheet. One use of this code sample is that you can generate a spreadsheet with the IDOC schema, and then you can map this schema to an Industry Standard XML schema or DTD (for example RosettaNet PIPs or CIDX Interfaces).

By: Yukai (Steven) Shi

Company: SAP Labs

Date: 06 October 2004

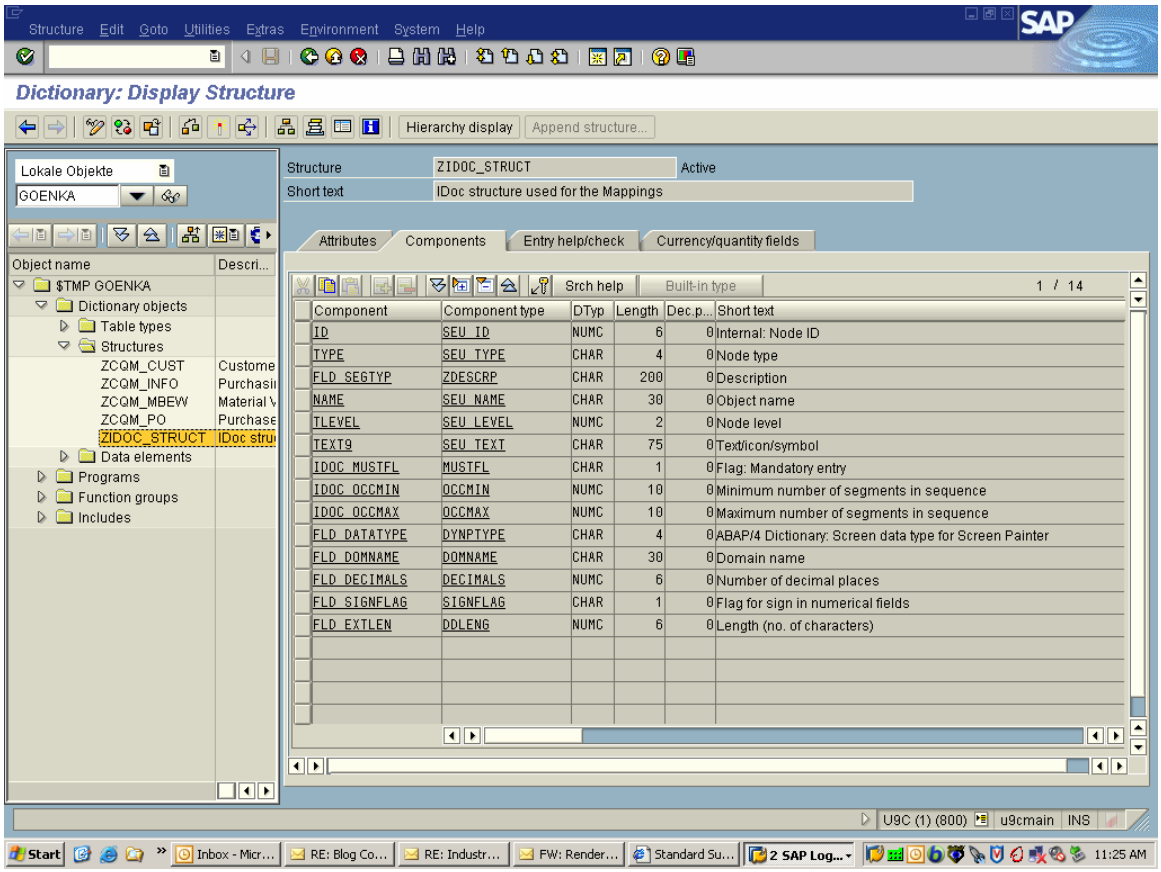
Instructions:

First you will need to create this local structure called ZIDOC_STRUCT in TCode SE11.

Then you will need to copy the program into your local space in R/3. Then create and add the INCLUDEs and the Function Modules that follow (in TCode SE80).

Note: if this document is in PDF format, you should be able to copy-paste the code.

Structure of ZIDOC_STRUCT


Structure Edit Goto Utilities Extras Environment System Help
Dictionary: Display Structure
Hierarchy display
Append structure...
Lokale Objekte
GOENKA
Object name
Descri...
$TMP GOENKA
Dictionary objects
Table types
Structures
ZCQM_CUST
ZCQM_INFO
ZCQM_MBEW
ZCQM_PO
ZIDOC_STRUCT
Data elements
Programs
Function groups
Includes
Structure ZIDOC_STRUCT Active
Short text IDoc structure used for the Mappings
Attributes
Components
Entry help/check
Currency/quantity fields
Srch help
Built-in type
1 / 14
Component | Component type | DTyp | Length | Dec.p... | Short text
ID | SEU_ID | NUMC | 6 | 0 | Internal: Node ID
TYPE | SEU_TYPE | CHAR | 4 | 0 | Node type
FLD_SEGTYP | ZDESCRP | CHAR | 200 | 0 | Description
NAME | SEU_NAME | CHAR | 30 | 0 | Object name
TLEVEL | SEU_LEVEL | NUMC | 2 | 0 | Node level
TEXT9 | SEU_TEXT | CHAR | 75 | 0 | Text/icon/symbol
IDOC_MUSTFL | MUSTFL | CHAR | 1 | 0 | Flag: Mandatory entry
IDOC_OCCMIN | OCCMIN | NUMC | 10 | 0 | Minimum number of segments in sequence
IDOC_OCCMAX | OCCMAX | NUMC | 10 | 0 | Maximum number of segments in sequence
FLD_DATATYPE | DYNPTYPE | CHAR | 4 | 0 | ABAP/4 Dictionary: Screen data type for Screen Painter
FLD_DOMNAME | DOMNAME | CHAR | 30 | 0 | Domain name
FLD_DECIMALS | DECIMALS | NUMC | 6 | 0 | Number of decimal places
FLD_SIGNFLAG | SIGNFLAG | CHAR | 1 | 0 | Flag for sign in numerical fields
FLD_EXTLEN | DDLENG | NUMC | 6 | 0 | Length (no. of characters)
U9C (1) (800)
u9cmain
INS
Start
Inbox - Micr...
RE: Blog Co...
RE: Industr...
FW: Render...
Standard Su...
2 SAP Log...
11:25 AM

```
*&---------------------------------------------------------------------*
*& Module pool       Z_RIDOCSHOW                                       *
*&                                                                     *
*&---------------------------------------------------------------------*
*&                                                                     *
*&                                                                     *
*&---------------------------------------------------------------------*


INCLUDE z_ridocshowtop.

INCLUDE z_ridocshowf01.

INITIALIZATION.

AT SELECTION-SCREEN ON sc_idoc.


START-OF-SELECTION.

  REFRESH sc_idoc.

  sed5struc-object = sc_idoc-low.
  sed5struc-select_org = p_basic.
  sed5struc-select_ext = p_ext.
  IF sed5struc-object IS INITIAL.
    MESSAGE e001(idmc).
    EXIT.
  ENDIF.


  IF NOT sed5struc-select_org IS INITIAL.
    DATA : l_idoctype LIKE edi_iapi00-idoctyp.

    l_idoctype = sed5struc-object.

    CALL FUNCTION 'IDOCTYPE_EXISTENCE_CHECK'
         EXPORTING
              pi_idoctyp       = l_idoctype
         EXCEPTIONS
              object_not_found = 1
              db_error         = 2
              OTHERS           = 3.
    CASE sy-subrc.
      WHEN 0.   " ok
      WHEN OTHERS.
        MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno WITH
            sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
        EXIT.
    ENDCASE.
* does the user request an extension type
  ELSEIF NOT sed5struc-select_ext IS INITIAL.
```

```
    DATA: l_cimtype  LIKE edi_iapi00-cimtyp.
    l_cimtype = sed5struc-object.

    CALL FUNCTION 'EXTTYPE_EXISTENCE_CHECK'
         EXPORTING
              pi_cimtyp        = l_cimtype
         EXCEPTIONS
              object_not_found = 1
              db_error         = 2
              OTHERS           = 3.
    CASE sy-subrc.
      WHEN 0.   " ok
      WHEN OTHERS.
        MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno WITH
            sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
        EXIT.
    ENDCASE.
  ENDIF.

END-OF-SELECTION.


  DATA: l_objecttype TYPE c,
        l_obj        TYPE edi_iapi00-idoctyp,
        released     LIKE sy-saprl,
        version      LIKE edi_verrec-version.

  released = sy-saprl.
  version  = '3'.
  l_obj = sed5struc-object.

  IF NOT sed5struc-select_ext IS INITIAL.
    l_objecttype = 'E'.
  ELSEIF NOT sed5struc-select_org IS INITIAL.
    l_objecttype = 'B'.
  ENDIF.

*=======================
  DATA: l_tree TYPE snodetext OCCURS 1,
        lt_final TYPE zt_idoc_struct.

  CALL FUNCTION 'ZEDI_FILL_IDOC_TREE'
       EXPORTING
            object                = l_obj
            object_type           = l_objecttype
            object_release        = released
            pi_rec_version        = version
            enable_further_output = 'X'
       TABLES
            pt_idocstruct         = lt_final
       EXCEPTIONS
            OTHERS                = 1.
  IF sy-subrc NE 0.
```

```
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH
        sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.

*=========================

  CONCATENATE l_idoctype '.xls' INTO l_idoctype.

  CALL FUNCTION 'WS_EXCEL'
       EXPORTING
            filename = l_idoctype
            synchron = ' '
       TABLES
            data     = lt_final.

*====================================
*  END OF PROGRAM
*====================================
```

```
*---------------------------------------------------------------------*
***INCLUDE Z_RIDOCSHOWF01 .
*---------------------------------------------------------------------*
*&---------------------------------------------------------------------*
*&      Form  initialize
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
FORM initialize.

ENDFORM.                    " initialize
*&---------------------------------------------------------------------*
*&      Form  read_user_input
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*      <--P_LT_IDOC  text
*---------------------------------------------------------------------*
FORM read_user_input CHANGING PT_IDOC.

ENDFORM.                    " read_user_input
```

```
*&---------------------------------------------------------------------*
*& Include Z_RIDOCSHOWTOP                                              *
*&                                                                     *
*&---------------------------------------------------------------------*

PROGRAM  Z_RIDOCSHOW.

TABLES: sed5struc.

TYPE-POOLS: LEDID.

TYPES: BEGIN OF ty_output,
         ID LIKE snodetext-id,
         name LIKE snodetext-name,
         tlevel LIKE snodetext-tlevel,
         text8 LIKE snodetext-text8,
         tlength8 LIKE snodetext-tlength8,
         text9 LIKE snodetext-text9,
         tlength9 LIKE snodetext-tlength9,
       END OF ty_output.

*---------------------------------------------------------------------*
* selections                                                          *
*---------------------------------------------------------------------*
SELECTION-SCREEN BEGIN OF BLOCK selection WITH FRAME TITLE text-011.
SELECT-OPTIONS sc_idoc FOR sed5struc-object NO INTERVALS NO-EXTENSION
               OBLIGATORY MEMORY ID idoc.
SELECTION-SCREEN END OF BLOCK selection.

*--- options
SELECTION-SCREEN BEGIN OF BLOCK control
                 WITH FRAME TITLE text-012.
*PARAMETERS: p_delta TYPE dis_delta DEFAULT ' ' AS CHECKBOX.
PARAMETERS: p_basic  LIKE SED5STRUC-SELECT_ORG DEFAULT 'X'.
PARAMETERS: p_ext    LIKE SED5STRUC-SELECT_EXT.
* parameters: p_batch like lko74-batch.

SELECTION-SCREEN END OF BLOCK control.
```

```
FUNCTION zedi_fill_idoc_tree.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     VALUE(OBJECT) TYPE  LEDID_IDOCTYPE
*"     VALUE(OBJECT_TYPE) TYPE  LEDID_STRUCT_TYPE
*"     VALUE(OBJECT_RELEASE) LIKE  SY-SAPRL DEFAULT SY-SAPRL
*"     VALUE(PI_REC_VERSION) LIKE  EDI_VERREC-VERSION DEFAULT '3'
*"     VALUE(ENABLE_FURTHER_OUTPUT) TYPE  LEDID_FLAG DEFAULT 'X'
*"  TABLES
*"      PT_IDOCSTRUCT TYPE  ZT_IDOC_STRUCT
*"----------------------------------------------------------------------

  TYPES: BEGIN OF ty_pre,
           tlevel   TYPE i,
           pre(150) TYPE c,
         END OF ty_pre.

  DATA: ls_idocstruct  TYPE zidoc_struct,
        ls_tree        TYPE snodetext,
        ls_idoc        TYPE ledid_idoc_struct,
        ls_segment     TYPE ledid_segment,
        ls_fields      TYPE ledid_segment_struct,
        ls_pre         TYPE ty_pre,
        lt_pre         TYPE STANDARD TABLE OF ty_pre,
        l_pre(200)     TYPE c,
        l_index        TYPE i.

* initialize global tables
  REFRESH: t_idoc,
           t_segments,
           t_fields.

  CALL FUNCTION 'IDOC_TYPE_COMPLETE_READ'
       EXPORTING
            struct_type    = object_type
            idoctype       = object
            release        = object_release
            version        = pi_rec_version
       IMPORTING
            idoc_type      = idoc
       TABLES
            idoc_struct    = t_idoc
            segments       = t_segments
            segment_struct = t_fields
       EXCEPTIONS
            OTHERS         = 1.
  CASE sy-subrc.
    WHEN 0.
*     fill global variable G_OBJECT_RELEASE
      g_object_release = object_release.

*     fill global variable G_OBJECT_VERSION
```

```
      g_object_version = pi_rec_version.

*     an IDoc type is now displayed
      g_mode = c_idoc_type.

*     initialize global tables and data
      REFRESH t_excl.

      CALL FUNCTION 'ZEDI_CONVERT_IDOC_TAB_TO_TREE'
           TABLES
                tree = tree.


      CALL FUNCTION 'RS_TREE_CONSTRUCT'
           TABLES
                nodetab = tree.

      SORT: t_idoc     BY segment_type,
            t_segments BY segment_type,
            t_fields   BY segment_type  fieldname.

      ls_idocstruct-id = 'ID'.
      ls_idocstruct-FLD_SEGTYP = 'Path'.
      ls_idocstruct-name = 'Name'.
      ls_idocstruct-tlevel = 'Depth'.
      ls_idocstruct-idoc_mustfl = 'Mandatory'.
      ls_idocstruct-idoc_occmin = 'Seg_OccMin'.
      ls_idocstruct-idoc_occmax = 'Seg_occMax'.
      ls_idocstruct-fld_datatype = 'Type'.
      ls_idocstruct-fld_domname = 'DomName'.
      ls_idocstruct-fld_decimals = 'DECIML'.
      ls_idocstruct-fld_signflag = 'Sign'.
      ls_idocstruct-fld_extlen = 'Len'.
      ls_idocstruct-text9 = 'Text'.

      APPEND ls_idocstruct TO pt_idocstruct.
      CLEAR: ls_idocstruct.

      LOOP AT tree INTO ls_tree.
        MOVE-CORRESPONDING ls_tree TO ls_idocstruct.

        IF ls_tree-type = 'TB'.

          CLEAR ls_pre.

          IF ls_tree-tlevel = 2.
            REFRESH lt_pre.
            CLEAR l_pre.
            ls_pre-tlevel = 1.
            APPEND ls_pre TO lt_pre.
          ENDIF.

          IF ls_tree-tlevel = 2.
```

```
          l_pre = ls_tree-name.
        ELSEIF ls_tree-tlevel > 2.
          l_index = ls_tree-tlevel - 1.
          READ TABLE lt_pre INTO ls_pre INDEX l_index.
          CONCATENATE ls_pre-pre ' /' ls_tree-name INTO l_pre.
        ENDIF.

        ls_pre-tlevel = ls_tree-tlevel.
        ls_pre-pre    = l_pre.
        INSERT ls_pre INTO lt_pre INDEX ls_tree-tlevel.

        READ TABLE t_idoc INTO ls_idoc
             WITH KEY segment_type = ls_tree-name
             BINARY SEARCH.
        IF sy-subrc IS INITIAL.
          ls_idocstruct-idoc_mustfl = ls_idoc-syntax_attrib-mustfl.
          ls_idocstruct-idoc_occmin = ls_idoc-syntax_attrib-occmin.
          ls_idocstruct-idoc_occmax = ls_idoc-syntax_attrib-occmax.
        ENDIF.

      ELSEIF ls_tree-type = 'DE'.

        READ TABLE t_fields INTO ls_fields
             WITH KEY segment_type = ls_tree-text7
                      fieldname    = ls_tree-name
             BINARY SEARCH.
        IF sy-subrc IS INITIAL.
         ls_idocstruct-fld_decimals = ls_fields-field_attrib-decimals.
         ls_idocstruct-fld_signflag = ls_fields-field_attrib-signflag.
          ls_idocstruct-fld_extlen = ls_fields-field_attrib-extlen.
          ls_idocstruct-fld_domname = ls_fields-field_attrib-domname.
         ls_idocstruct-fld_datatype = ls_fields-field_attrib-datatype.
        ENDIF.

      ENDIF.

      ls_idocstruct-fld_segtyp = l_pre.

      APPEND ls_idocstruct TO pt_idocstruct.
      CLEAR: ls_idoc, ls_segment, ls_fields, ls_idocstruct.
    ENDLOOP.

  WHEN OTHERS.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDCASE.
ENDFUNCTION.
```

```
FUNCTION ZEDI_CONVERT_IDOC_TAB_TO_TREE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       TABLES
*"              TREE STRUCTURE  SNODETEXT
*"----------------------------------------------------------------------

  DATA: WA_IDOC         TYPE LEDID_IDOC_STRUCT.

  REFRESH: TREE.

* IDOC contains the object type: basis idoc type, extension type
* or idoc type.

* get user specific attribute setting
*  perform fa2_get_user_attributes  using edi_attrib.
  PERFORM FA2_GET_USER_ATTRIBUTES_ALL.

* construct tree based on idoc type data and flags
  PERFORM F01_CONSTRUCT_ROOT_NODE TABLES   TREE
                                  CHANGING EDI_ATTRIB.

* construct every other node
  LOOP AT T_IDOC INTO WA_IDOC.
* new reference segment starts here? insert dummy node, set level to 2
*   IF last_refsegment NE wa_idoc-ref_segment.
*     last_refsegment = wa_idoc-ref_segment.
*    PERFORM f01_construct_refsegment_node TABLES tree
*                                          USING wa_idoc.
*   ENDIF.

    PERFORM F01_CONSTRUCT_SEGMENT_NODE TABLES TREE
                                       USING  WA_IDOC
                                              EDI_ATTRIB.

* adjust attributes in IDoc table according to L_ATTRIBUTES
    IF EDI_ATTRIB-DO2 EQ ON.
      WA_IDOC-DONT_SHOW_DOCU = OFF.
    ELSE.
      WA_IDOC-DONT_SHOW_DOCU = ON.
    ENDIF.
    IF EDI_ATTRIB-AT2 EQ ON.
      WA_IDOC-DONT_SHOW_ATTRIB = OFF.
    ELSE.
      WA_IDOC-DONT_SHOW_ATTRIB = ON.
    ENDIF.
    MODIFY T_IDOC FROM WA_IDOC.

  ENDLOOP.

ENDFUNCTION.
```

```
FUNCTION ZDOWNLOAD_TABLE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  TABLES
*"      T_TEXT STRUCTURE  LINE
*"  CHANGING
*"     REFERENCE(P_FILE) LIKE  RLGRAP-FILENAME
*"     REFERENCE(P_PATH) LIKE  RLGRAP-FILENAME
*"     REFERENCE(P_RETURNCODE) LIKE  SY-SUBRC
*"----------------------------------------------------------------------

  DATA : CURR_DIR           LIKE RLGRAP-FILENAME,
         COMPLETE_FILENAME  LIKE RLGRAP-FILENAME,
         L_RC LIKE SY-SUBRC,
         L_LINE(180),       " text + complete_filename(128)
         L_ANSWER(1).
** is path given ?
  IF P_PATH NE SPACE.
** check the existence of the given directory
    CALL FUNCTION 'WS_QUERY'
         EXPORTING
*           ENVIRONMENT    = ' '
            FILENAME       = P_PATH
            QUERY          = 'DE'
*           WINID          = ' '
         IMPORTING
              RETURN         = L_RC
         EXCEPTIONS
              OTHERS         = 1.
    CASE SY-SUBRC.
      WHEN 0.
        IF L_RC EQ 0.
** no such directory -> clear path
          CLEAR P_PATH.
        ENDIF.
      WHEN OTHERS.
** could not query
        P_RETURNCODE = 4.
    ENDCASE.
  ENDIF.

  IF P_PATH EQ SPACE.
** get the actual working directory
   CALL FUNCTION 'WS_QUERY'
       EXPORTING
*           ENVIRONMENT    = ' '
*           FILENAME       = ' '
            QUERY          = 'CD'
*           WINID          = ' '
       IMPORTING
            RETURN         = CURR_DIR
       EXCEPTIONS
            OTHERS         = 1.
```

```
   CASE SY-SUBRC.
    WHEN 0.
** no valid path so far? -> use the working directory
       P_PATH = CURR_DIR.
    WHEN OTHERS.
** we have a problem here
      P_RETURNCODE = 4.
   ENDCASE.
  ENDIF.
** ---------------------------------------------------------------------
** our destination file must be given !
  IF P_FILE EQ SPACE.
    P_RETURNCODE = 4.
  ELSE.
** generate filename
    CONCATENATE P_PATH P_FILE INTO COMPLETE_FILENAME.
** check if file exists and if so start dialog
*    IF P_WITH_CHECK NE SPACE.
      CALL FUNCTION 'WS_QUERY'
           EXPORTING
*               ENVIRONMENT    =
                FILENAME       = COMPLETE_FILENAME
                 QUERY          = 'FE'
*               WINID          =
          IMPORTING
                RETURN         = L_RC
           EXCEPTIONS
                 OTHERS         = 1.
      IF SY-SUBRC EQ 0.
        IF L_RC NE 0.                   " file exists already
          L_LINE = TEXT-255.
          REPLACE '$' WITH COMPLETE_FILENAME INTO L_LINE.
          CONDENSE L_LINE.
          CALL FUNCTION 'POPUP_TO_CONFIRM_STEP'
               EXPORTING
                   TEXTLINE1      = L_LINE
                   TEXTLINE2      = TEXT-256
                   TITEL          = TEXT-257
                    CANCEL_DISPLAY = ' '
             IMPORTING
                   ANSWER         = L_ANSWER
               EXCEPTIONS
                    OTHERS         = 0.
          IF L_ANSWER NE 'J'.
            SY-MSGID = 'EA'.
            SY-MSGTY = 'I'.
            SY-MSGNO = 320.
            P_RETURNCODE = 4.
            EXIT.
          ENDIF.
        ENDIF.
      ENDIF.
*    ENDIF.
```

```
** do the download
    CALL FUNCTION 'WS_DOWNLOAD'
         EXPORTING
              FILENAME            = COMPLETE_FILENAME
              FILETYPE            = 'ASC'
         TABLES
              DATA_TAB            = T_TEXT
*               FIELDNAMES          =
         EXCEPTIONS
              FILE_OPEN_ERROR     = 1
              FILE_WRITE_ERROR    = 2
              OTHERS              = 3.
    CASE SY-SUBRC.
      WHEN 0.
      WHEN 1.
        MESSAGE I321 WITH P_FILE P_PATH.
        P_RETURNCODE = 6.
      WHEN 2.
        MESSAGE I322 WITH P_FILE P_PATH.
** error when downloading
        P_RETURNCODE = 7.
      WHEN OTHERS.
        MESSAGE I323 WITH P_FILE P_PATH.
        P_RETURNCODE = 8.
    ENDCASE.
  ENDIF.



ENDFUNCTION.
```

```
FUNCTION ZEDI_CONVERT_IDOC_REC_TO_TREE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(VERSION) LIKE  EDI_VERREC-VERSION DEFAULT '2'
*"       TABLES
*"              TREE STRUCTURE  SNODETEXT
*"----------------------------------------------------------------------

  DATA: L_IDOC_REC_STRUCT  TYPE LEDID_IDOC_RECORD_STRUCT,
        L_IDOC_REC         TYPE LEDID_IDOC_RECORD.

  REFRESH: TREE.

* get user specific attribute setting
*  perform fa2_get_user_attributes  using edi_attrib.
  PERFORM FA2_GET_USER_ATTRIBUTES_ALL.

* construct tree based on idoc type data and flags
  PERFORM F06_CONSTRUCT_REC_ROOT_NODE TABLES   TREE.

  LOOP AT T_IDOC_REC INTO L_IDOC_REC.

    PERFORM F06_CONSTRUCT_REC_NODE  TABLES  TREE
                                    USING   L_IDOC_REC
                                            VERSION
                                            EDI_ATTRIB.

* construct field nodes for the control record
    LOOP AT T_IDOC_REC_STRUCT INTO L_IDOC_REC_STRUCT
                WHERE RECORD_TYPE EQ L_IDOC_REC-RECORD_TYPE.

      PERFORM F06_CONSTRUCT_REC_FIELD_NODE TABLES TREE
                                           USING  L_IDOC_REC_STRUCT
                                                  VERSION
                                                  EDI_ATTRIB.

* adjust attributes in structure table according to EDI_ATTRIB
      L_IDOC_REC_STRUCT-DONT_SHOW_DOCU = EDI_ATTRIB-DO3.
      M_TOGGLE L_IDOC_REC_STRUCT-DONT_SHOW_DOCU.
      L_IDOC_REC_STRUCT-DONT_SHOW_ATTRIB = EDI_ATTRIB-AT3.
      M_TOGGLE L_IDOC_REC_STRUCT-DONT_SHOW_ATTRIB.
      L_IDOC_REC_STRUCT-DONT_SHOW_VALUES = EDI_ATTRIB-VAL.
      M_TOGGLE L_IDOC_REC_STRUCT-DONT_SHOW_VALUES.
      MODIFY T_IDOC_REC_STRUCT FROM L_IDOC_REC_STRUCT.

    ENDLOOP.

* adjust attribute/documentation flag
    L_IDOC_REC-DONT_SHOW_DOCU = EDI_ATTRIB-DO2.
    M_TOGGLE L_IDOC_REC-DONT_SHOW_DOCU.
    L_IDOC_REC-DONT_SHOW_ATTRIB = EDI_ATTRIB-AT2.
    M_TOGGLE L_IDOC_REC-DONT_SHOW_ATTRIB.
```

```
    MODIFY T_IDOC_REC FROM L_IDOC_REC.
  ENDLOOP.
ENDFUNCTION.
```

```
FUNCTION ZEDI_CONVERT_SEGMENT_TO_TREE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       TABLES
*"              TREE STRUCTURE  SNODETEXT
*"----------------------------------------------------------------------
  data: wa_idoc         type ledid_idoc_struct.

* get user specific attribute setting
  perform fa2_get_user_attributes_all.

  loop at t_idoc into wa_idoc where segment_type eq segment.
* fill wa_idoc, only 1 record in t_idoc
    wa_idoc-dont_show_attrib = edi_attrib-at2.
    m_toggle wa_idoc-dont_show_attrib.
    wa_idoc-dont_show_docu = edi_attrib-do2.
    m_toggle wa_idoc-dont_show_docu.
    modify t_idoc from wa_idoc.
  endloop.
  refresh: tree.
  clear tree.
  tree-type      = type_segment.
  tree-tlevel    = 1.
* fill segment information into TREE-NAME
  tree-name      = wa_idoc-segment_type.
  tree-nlength   = length_segment.
  tree-color     = col_segment.

* show description in the last text field
  perform fa2_fill_field using tree on '9' c_text
                               wa_idoc-segment_type_attrib-descrp.

* add checkbox for attributes
  perform fa1_add_checkbox  using tree
                                  edi_attrib-at2
                                  edi_attrib-do2
                                  dont_display.

  append tree.

  perform f01_construct_field_node tables tree
                              using wa_idoc-segment_type
                              tree-tlevel
                              edi_attrib.


endfunction.
```

```
FUNCTION ZEDI_CREATE_SYIDOC01.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(OBJECT_RELEASE) LIKE  SY-SAPRL DEFAULT SY-SAPRL
*"       EXPORTING
*"             VALUE(DOCNUM) LIKE  EDIDC-DOCNUM
*"----------------------------------------------------------------------

  DATA: T_EDIDD  LIKE EDIDD OCCURS 1 WITH HEADER LINE,
        L_EDIDC  LIKE EDIDC,
        L_ID     LIKE EDIDC-DOCNUM,
        L_EDK13  LIKE EDK13,
        L_EDD13  LIKE EDD13.

* display settings for changes
  CALL FUNCTION 'IDOC_DOC_IDOC_OUT_DISPLAY'
      EXPORTING
           TITLE_POPUP      = TEXT-304
      CHANGING
           DEF_PARNUM       = EDI_IDCATT-PARNUM
           DEF_PARTYP       = EDI_IDCATT-PARTYP
       EXCEPTIONS
            SELECTION_CANCEL = 1
            OTHERS           = 2.
 CASE SY-SUBRC.
  WHEN 0.
* read partner agreement for PARNUM/PARTYP
  L_EDK13-RCVPRN = EDI_IDCATT-PARNUM.
  L_EDK13-RCVPRT = EDI_IDCATT-PARTYP.
  L_EDK13-MESTYP = C_MESTYP_SYIDOC.

  CALL FUNCTION 'EDI_AGREE_OUT_MESSTYPE_READ'
       EXPORTING
            REC_EDK13 = L_EDK13
       IMPORTING
            REC_EDD13 = L_EDD13.

* test partner agreement for syidoc01!

  L_EDIDC-RCVPRN = L_EDK13-RCVPRN.
  L_EDIDC-RCVPRT = L_EDK13-RCVPRT.
  PERFORM FA1_GET_LOGICAL_SYSTEM USING L_EDIDC-SNDPRN.
  L_EDIDC-SNDPRT = C_LOGICAL_SYSTEM.
* l_edidc-doctyp = 'SYIDOC01'.
  L_EDIDC-IDOCTP = L_EDD13-IDOCTYP.
  L_EDIDC-MESTYP = C_MESTYP_SYIDOC.
  L_EDIDC-DIRECT = '1'.
  L_EDIDC-OUTMOD = L_EDD13-OUTMOD.
  L_EDIDC-RCVPOR = L_EDD13-RCVPOR.
  L_EDIDC-STD    = L_EDD13-STD.
  L_EDIDC-STDVRS = L_EDD13-STDVRS.
  L_EDIDC-STDMES = L_EDD13-STDMES.
```

```
  CALL FUNCTION 'EDI_FILL_SYIDOC01'
       EXPORTING
            OBJECT_RELEASE = OBJECT_RELEASE
            PI_EDIVIEW     = L_EDD13-EDIVIEW
       TABLES
            T_EDIDD        = T_EDIDD
       CHANGING
            IDOC_CONTROL   = L_EDIDC.

  CALL FUNCTION 'EDI_DOCUMENT_OPEN_FOR_CREATE'
       EXPORTING
            IDOC_CONTROL = L_EDIDC
       IMPORTING
            IDENTIFIER   = L_ID.

  CALL FUNCTION 'EDI_SEGMENTS_ADD_BLOCK'
       EXPORTING
            IDENTIFIER      = L_ID
       TABLES
            IDOC_CONTAINERS = T_EDIDD.

  CALL FUNCTION 'EDI_DOCUMENT_CLOSE_CREATE'
       EXPORTING
            IDENTIFIER   = L_ID
       IMPORTING
            IDOC_CONTROL = L_EDIDC.

  DOCNUM = L_EDIDC-DOCNUM.

* set state 'ready for dispatching'
  PERFORM FA4_SET_EDI_STATUS  USING DOCNUM
                                    C_READY_FOR_DISPATCHING.

  MESSAGE S097 WITH DOCNUM.

* send IDoc (according to OUTMOD)
  PERFORM FA4_SEND_IDOC   USING DOCNUM
                                L_EDD13-OUTMOD.
 WHEN '1'.
   MESSAGE I571.
 WHEN OTHERS.
 ENDCASE.
ENDFUNCTION.
```

```
FUNCTION ZEDI_CREATE_SYRECD01.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(VERSION) LIKE  EDI_VERREC-VERSION DEFAULT '2'
*"       EXPORTING
*"             VALUE(DOCNUM) LIKE  EDIDC-DOCNUM
*"----------------------------------------------------------------------

  DATA: T_EDIDD  LIKE EDIDD OCCURS 1 WITH HEADER LINE,
        L_EDIDC  LIKE EDIDC,
        L_ID     LIKE EDIDC-DOCNUM,
        L_EDK13  LIKE EDK13,
        L_EDD13  LIKE EDD13.

* display settings for changes
  CALL FUNCTION 'IDOC_DOC_IDOC_OUT_DISPLAY'
       EXPORTING
            TITLE_POPUP      = TEXT-304
       CHANGING
            DEF_PARNUM       = EDI_IDCATT-PARNUM
            DEF_PARTYP       = EDI_IDCATT-PARTYP
       EXCEPTIONS
            SELECTION_CANCEL = 1
            OTHERS           = 2.
  CASE SY-SUBRC.
    WHEN 0.
* read partner agreement for PARNUM/PARTYP
      L_EDK13-RCVPRN = EDI_IDCATT-PARNUM.
      L_EDK13-RCVPRT = EDI_IDCATT-PARTYP.
      L_EDK13-MESTYP = C_MESTYP_SYRECD.

      CALL FUNCTION 'EDI_AGREE_OUT_MESSTYPE_READ'
           EXPORTING
                REC_EDK13 = L_EDK13
           IMPORTING
                REC_EDD13 = L_EDD13.

      L_EDIDC-RCVPRN = L_EDK13-RCVPRN.
      L_EDIDC-RCVPRT = L_EDK13-RCVPRT.
      PERFORM FA1_GET_LOGICAL_SYSTEM USING L_EDIDC-SNDPRN.
      L_EDIDC-SNDPRT = C_LOGICAL_SYSTEM.
      L_EDIDC-IDOCTP = L_EDD13-IDOCTYP.
      L_EDIDC-MESTYP = C_MESTYP_SYRECD.
      L_EDIDC-DIRECT = '1'.
      L_EDIDC-OUTMOD = L_EDD13-OUTMOD.
      L_EDIDC-RCVPOR = L_EDD13-RCVPOR.
      L_EDIDC-STD    = L_EDD13-STD.
      L_EDIDC-STDVRS = L_EDD13-STDVRS.
      L_EDIDC-STDMES = L_EDD13-STDMES.

      CALL FUNCTION 'EDI_FILL_SYRECD01'
           EXPORTING
```

```
               VERSION      = VERSION
               PI_EDIVIEW   = L_EDD13-EDIVIEW
          TABLES
               T_EDIDD      = T_EDIDD
          CHANGING
               IDOC_CONTROL = L_EDIDC.

     CALL FUNCTION 'EDI_DOCUMENT_OPEN_FOR_CREATE'
          EXPORTING
               IDOC_CONTROL = L_EDIDC
          IMPORTING
               IDENTIFIER   = L_ID.

     CALL FUNCTION 'EDI_SEGMENTS_ADD_BLOCK'
          EXPORTING
               IDENTIFIER      = L_ID
          TABLES
               IDOC_CONTAINERS = T_EDIDD.

     CALL FUNCTION 'EDI_DOCUMENT_CLOSE_CREATE'
          EXPORTING
               IDENTIFIER   = L_ID
          IMPORTING
               IDOC_CONTROL = L_EDIDC.

     DOCNUM = L_EDIDC-DOCNUM.

* set state 'ready for dispatching'
     PERFORM FA4_SET_EDI_STATUS  USING DOCNUM
                                       C_READY_FOR_DISPATCHING.

     MESSAGE S097 WITH DOCNUM.

* send IDoc (according to OUTMOD)
     PERFORM FA4_SEND_IDOC   USING DOCNUM
                                   L_EDD13-OUTMOD.
   WHEN 1.
     MESSAGE I571.
   WHEN OTHERS.
     MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
             WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
 ENDCASE.
ENDFUNCTION.
```

```
FUNCTION ZEDI_FILL_SYIDOC01.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"            VALUE(OBJECT_RELEASE) LIKE  EDBAS-RELEASED
*"          DEFAULT SY-SAPRL
*"            VALUE(PI_EDIVIEW) LIKE  EDVIEW-EDIVIEW OPTIONAL
*"       TABLES
*"             T_EDIDD STRUCTURE  EDIDD
*"       CHANGING
*"            VALUE(IDOC_CONTROL) LIKE  EDIDC STRUCTURE  EDIDC
*"       EXCEPTIONS
*"             WRONG_BASIS_IDOC_TYPE
*"----------------------------------------------------------------------

  DATA: WA_IDOC    TYPE LEDID_IDOC_STRUCT,
        WA_SEGMENT TYPE LEDID_SEGMENT,
        WA_FIELD   TYPE LEDID_SEGMENT_STRUCT,
        SEGM_FLAG(1) TYPE C,
        VIEW_FLAG(1) TYPE C,
        LT_VIEWSYN LIKE VIEWSYN OCCURS 0 WITH HEADER LINE.

  IF IDOC_CONTROL-IDOCTP NE 'SYIDOC01'.
    MESSAGE E572 WITH 'SYIDOC01' IDOC_CONTROL-IDOCTP
                 RAISING WRONG_BASIS_IDOC_TYPE.
  ENDIF.

* new since 4.5A: read view - if requested
  IF NOT PI_EDIVIEW IS INITIAL.
    PERFORM FA5_READ_VIEW TABLES  LT_VIEWSYN
                          USING   PI_EDIVIEW.
    VIEW_FLAG = ON.
  ELSE.
    VIEW_FLAG = OFF.
  ENDIF.

  REFRESH T_EDIDD.

*** CHECK AND FILL E1IDOCH
* if view selected: segment E1IDOCH in view?
  PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                           USING 'E1IDOCH' SEGM_FLAG VIEW_FLAG.
  IF SEGM_FLAG EQ ON.
    PERFORM FA3_FILL_E1IDOCH TABLES T_EDIDD
                             USING  IDOC_CONTROL
                                    SY-LANGU
                                    OBJECT_RELEASE.
  ENDIF.

*** CHECK AND FILL E1IDOCT
* if view selected: segment E1IDOCT in view?
  PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                           USING 'E1IDOCT' SEGM_FLAG VIEW_FLAG.
```

```
  IF SEGM_FLAG EQ ON.
    PERFORM FA3_FILL_E1IDOCT TABLES T_EDIDD
                             USING  IDOC_CONTROL.
  ENDIF.

  IF IDOC-DONT_SHOW_DOCU EQ OFF.
* no documentation yet for IDoc types
  ENDIF.

*** CHECK AND FILL E1SEGMH
* if view selected: segment E1SEGMH in view?
  PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                           USING 'E1SEGMH' SEGM_FLAG VIEW_FLAG.

  IF SEGM_FLAG EQ ON.
    LOOP AT T_IDOC INTO WA_IDOC.
      LOOP AT T_SEGMENTS INTO WA_SEGMENT
                         WHERE SEGMENT_TYPE EQ WA_IDOC-SEGMENT_TYPE.
* fill segment header segment
        PERFORM FA3_FILL_E1SEGMH  TABLES T_EDIDD
                                  USING  IDOC_CONTROL
                                         WA_IDOC
                                         WA_SEGMENT.
        IF WA_IDOC-DONT_SHOW_DOCU EQ OFF.
* if view selected: segment E1TXTH2 in view?
          PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                           USING 'E1TXTH2' SEGM_FLAG VIEW_FLAG.
          IF SEGM_FLAG EQ ON.
* get documentation for the current segment
* if view selected: segment E1TXTP2 in view?
            PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                           USING 'E1TXTP2' SEGM_FLAG VIEW_FLAG.
            PERFORM FA3_FILL_TXT2     TABLES T_EDIDD
                                      USING  IDOC_CONTROL
                                             WA_IDOC-SEGMENT_TYPE
                                             SEGM_FLAG.
          ENDIF.
        ENDIF.
        EXIT.
      ENDLOOP.

*** end of segments for segment-documentation
*** begin of segments for segment-structure

      DATA: L_FIELD_POS LIKE E1SEGMF-FIELD_POS.
      CLEAR L_FIELD_POS.

*** CHECK AND FILL E1SEGMF
* if view selected: segment E1SEGMF in view?
      PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                             USING 'E1SEGMF' SEGM_FLAG VIEW_FLAG.
      IF SEGM_FLAG EQ ON.
        LOOP AT T_FIELDS INTO WA_FIELD
```

```
                        WHERE SEGMENT_TYPE EQ WA_IDOC-SEGMENT_TYPE.
* fill field information for the current segment
         ADD 1 TO L_FIELD_POS.
         PERFORM FA3_FILL_E1SEGMF  TABLES T_EDIDD
                                   USING  IDOC_CONTROL
                                          L_FIELD_POS
                                          WA_FIELD
                                          WA_IDOC-SEGMENT_TYPE.

         IF WA_FIELD-DONT_SHOW_DOCU EQ OFF.
* if view selected: segment E1TXTH3 in view?
           PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                        USING 'E1TXTH3' SEGM_FLAG VIEW_FLAG.
           IF SEGM_FLAG EQ ON.
* get documentation for the current field
* if view selected: segment E1TXTP3 in view?
             PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                          USING 'E1TXTP3' SEGM_FLAG VIEW_FLAG.

             PERFORM FA3_FILL_TXT3  TABLES T_EDIDD
                                    USING  IDOC_CONTROL
                                     WA_FIELD-FIELD_ATTRIB-ROLLNAME
                                     SEGM_FLAG.
           ENDIF.
         ENDIF.
* append possible field values
         IF WA_FIELD-DONT_SHOW_VALUES EQ OFF.
* if view selected: segment E1VALU1 in view?
            PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                         USING 'E1VALU1' SEGM_FLAG VIEW_FLAG.
            IF SEGM_FLAG EQ ON.
                PERFORM FA3_FILL_E1VALU1  TABLES T_EDIDD
                                    USING  IDOC_CONTROL
                                           WA_IDOC-SEGMENT_TYPE
                                           WA_FIELD-FIELDNAME.
           ENDIF.
         ENDIF.
       ENDLOOP.
     ENDIF.      "if e1segmf
   ENDLOOP.
 ENDIF.          "if e1segmh

* add associated logical message types
* if view selected: segment E1MESH in view?
   PERFORM FA5_CHECK_VIEW TABLES LT_VIEWSYN
                          USING 'E1MESH' SEGM_FLAG VIEW_FLAG.
   IF SEGM_FLAG EQ ON.
      PERFORM FA3_FILL_E1MESH  TABLES T_EDIDD
                          USING  IDOC_CONTROL
                                 OBJECT_RELEASE.
   ENDIF.
ENDFUNCTION.
```

```
FUNCTION ZEDI_FILL_SYIDOC01_FOR_RFC.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(OBJECT) LIKE  EDI_IAPI00-DOCTYP
*"             VALUE(OBJECT_RELEASE) LIKE  SY-SAPRL DEFAULT SY-SAPRL
*"       TABLES
*"              T_EDIDD STRUCTURE  EDIDD_OLD
*"----------------------------------------------------------------------

* initialize global tables
  REFRESH: T_IDOC,
           T_SEGMENTS,
           T_FIELDS.

  CLEAR EDI_ATTRIB.

  DATA: L_OBJECT_TYPE TYPE LEDID_STRUCT_TYPE,
        L_OBJECT      TYPE LEDID_IDOCTYPE,
        L_IDOCTYP LIKE EDI_IAPI00-IDOCTYP,
        L_CIMTYP LIKE EDI_IAPI00-CIMTYP,
        L_EDIDC  LIKE EDIDC,
        LT_EDIDD LIKE EDIDD OCCURS 5 WITH HEADER LINE.

* basis idoc type or idoc type?
  CALL FUNCTION 'IDOC_GET_LONG_NAMES'
       EXPORTING
*           PI_IDOCTYP =
*           PI_CIMTYP  =
*           PI_MESTYP  =
            PI_DOCTYP  = OBJECT
       IMPORTING
            PE_IDOCTYP = L_IDOCTYP
            PE_CIMTYP  = L_CIMTYP
*           PE_MESTYP  =
       EXCEPTIONS
            OTHERS     = 0.

  IF L_CIMTYP EQ SPACE.
    L_OBJECT      = L_IDOCTYP.
    L_OBJECT_TYPE = LEDID_STRUCT_BASIS_IDOC.
  ELSE.
    L_OBJECT      = L_CIMTYP.
    L_OBJECT_TYPE = LEDID_STRUCT_EXTENSION.
  ENDIF.

  CALL FUNCTION 'IDOC_TYPE_COMPLETE_READ'
       EXPORTING
            STRUCT_TYPE    = L_OBJECT_TYPE
            IDOCTYPE       = L_OBJECT
            RELEASE        = OBJECT_RELEASE
       IMPORTING
            IDOC_TYPE      = IDOC
```

```
       TABLES
            IDOC_STRUCT    = T_IDOC
            SEGMENTS       = T_SEGMENTS
            SEGMENT_STRUCT = T_FIELDS
       EXCEPTIONS
            OTHERS         = 1.
  CASE SY-SUBRC.
    WHEN 0.

* get current configuration (will be used for documentation output)
      CALL FUNCTION 'IDOC_CONFIGURE_DOC'
           EXPORTING
                SERVICE       = C_PARAMETERS_READ
           CHANGING
                ATTRIB_DATA   = EDI_ATTRIB
                IDCATT_DATA   = EDI_IDCATT
                CFILE_DATA    = EDI_CFILE
                HTMLFILE_DATA = EDI_HTMFIL.

      PERFORM FA3_SET_ATTRIBUTES USING EDI_ATTRIB.

      L_EDIDC-IDOCTP = 'SYIDOC01'.

      CALL FUNCTION 'EDI_FILL_SYIDOC01'
           EXPORTING
                OBJECT_RELEASE = OBJECT_RELEASE
           TABLES
                T_EDIDD        = LT_EDIDD
           CHANGING
                IDOC_CONTROL   = L_EDIDC.

      PERFORM FA2_FILL_ADMIN_DATA  TABLES LT_EDIDD
                                   USING 'SYIDOC01'.

* convert internal structure EDIDD into pre-4.0 structure EDIDD_OLD
      LOOP AT LT_EDIDD.
        CLEAR T_EDIDD.
        MOVE-CORRESPONDING LT_EDIDD TO T_EDIDD.
        APPEND T_EDIDD.
      ENDLOOP.

* set current segment definition in table EDIDD
      PERFORM FA2_FILL_SEGTYP_DATA_OLD TABLES T_EDIDD.
    WHEN OTHERS.
      MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
                 WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDCASE.

ENDFUNCTION.
```

```
FUNCTION ZEDI_FILL_SYRECD01.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(VERSION) LIKE  EDI_VERREC-VERSION DEFAULT '2'
*"             VALUE(PI_EDIVIEW) LIKE  EDVIEW-EDIVIEW OPTIONAL
*"       TABLES
*"              T_EDIDD STRUCTURE  EDIDD
*"       CHANGING
*"             VALUE(IDOC_CONTROL) LIKE  EDIDC STRUCTURE  EDIDC
*"       EXCEPTIONS
*"              WRONG_BASIS_IDOC_TYPE
*"----------------------------------------------------------------------

  data:    wa_idoc_rec        type ledid_idoc_record,
           wa_idoc_rec_struct type ledid_idoc_record_struct,
           l_segment          like edisegmhd-segtyp,
           l_release          like sy-saprl,
           view_flag(1)       type c,
           segm_flag(1)       type c,
           lt_viewsyn like viewsyn occurs 0 with header line.

  refresh t_edidd.

  if idoc_control-idoctp ne 'SYRECD01' and
     idoc_control-idoctp ne 'SYRECD02'.
    message e572 with 'SYRECD01' idoc_control-idoctp
                 raising wrong_basis_idoc_type.
  endif.

* new since 4.5A: read view - if requested
  if not pi_ediview is initial.
    perform fa5_read_view tables  lt_viewsyn
                          using   pi_ediview.
    view_flag = on.
  else.
    view_flag  = off.
  endif.

  perform fa3_version_to_release using version
                                       l_release.

*** CHECK AND FILL E1IDOCH
* if view selected: segment E1IDOCH in view?
  perform fa5_check_view tables lt_viewsyn
                         using 'E1IDOCH' segm_flag view_flag.

* fill E1IDOCH
  if segm_flag eq on.
    perform fa3_fill_e1idoch tables t_edidd
                             using  idoc_control
                                    sy-langu
                                    l_release.
```

```
  endif.

***  CHECK AND FILL E1SEGMH
* if view selected: segment E1SEGMH in view?
  perform fa5_check_view tables lt_viewsyn
                         using 'E1SEGMH' segm_flag view_flag.

  if segm_flag eq on.
    loop at t_idoc_rec into wa_idoc_rec.
* fill record header segment
      perform fa3_fill_e1segmh_rec  tables t_edidd
                                    using  idoc_control
                                           version
                                           wa_idoc_rec.

      if wa_idoc_rec-dont_show_docu eq off.
***  CHECK AND FILL E1TXTH2
* if view selected: segment E1TXTH2 in view?
        perform fa5_check_view tables lt_viewsyn
                           using 'E1TXTH2' segm_flag view_flag.
        if segm_flag eq on.
* OK, fill e1txth2
* get documentation for the current record
* some type casting necessary
          l_segment = wa_idoc_rec-ddic_name.
*     PERFORM fa2_switch_id_to_name USING wa_idoc_rec-record_type
*                                         version
*                                         l_segment.

* check for segment E1TXTP2
* this segment will be filled (or not) in the following form
* if view selected: segment E1TXTP2 in view?
          perform fa5_check_view tables lt_viewsyn
                           using 'E1TXTP2' segm_flag view_flag.

          perform fa3_fill_txt2     tables t_edidd
                                    using  idoc_control
                                           l_segment
                                           segm_flag.
        endif.
      endif.

*** end of segment for segment-documentation  and
*** begin of segments for structure

      data: l_field_pos like e1segmf-field_pos.
      clear l_field_pos.

*** check and fill E1SEGMF
* if view selected: segment E1SEGMF in view?
      perform fa5_check_view tables lt_viewsyn
                             using 'E1SEGMF' segm_flag view_flag.
      if segm_flag eq on.
```

```
* consider all fields for the current IDoc record
       loop at t_idoc_rec_struct into wa_idoc_rec_struct
                          where record_type eq wa_idoc_rec-record_type.
* fill field information for the current record
         add 1 to l_field_pos.
         perform fa3_fill_e1segmf_rec  tables t_edidd
                                       using  idoc_control
                                              l_field_pos
                                              wa_idoc_rec_struct
                                              wa_idoc_rec.

         if wa_idoc_rec_struct-dont_show_docu eq off.
* if view selected: segment E1TXTH3 in view?
           perform fa5_check_view tables lt_viewsyn
                       using 'E1TXTH3' segm_flag view_flag.
           if segm_flag eq on.
* get documentation for the current field
* if view selected: segment E1TXTP3 in view?
             perform fa5_check_view tables lt_viewsyn
                         using 'E1TXTP3' segm_flag view_flag.
             perform fa3_fill_txt3     tables t_edidd
                                       using  idoc_control
                                  wa_idoc_rec_struct-field_attrib-rollname
                                                segm_flag.
           endif.
         endif.
* append possible field values
* some type casting necessary
         perform fa2_switch_id_to_name using wa_idoc_rec-record_type
                                             version
                                             l_segment.
         if wa_idoc_rec_struct-dont_show_values eq off.
* if view selected: segment E1VALU1/E1VALU2 in view?
           if idoc_control-idoctp eq 'SYPART02'.
            perform fa5_check_view tables lt_viewsyn
                                  using 'E1VALU2' segm_flag view_flag.
           else.
            perform fa5_check_view tables lt_viewsyn
                                   using 'E1VALU1' segm_flag view_flag.
           endif.
           if segm_flag eq on.
               perform fa3_fill_e1valu1  tables t_edidd
                                       using  idoc_control
                                              l_segment
                                     wa_idoc_rec_struct-fieldname.
           endif.
         endif.
       endloop.
     endif.                           " if E1SEGMF in view
   endloop.
 endif.                               "if E1SEGMH in view
endfunction.
```

```
FUNCTION ZEDI_FORMAT_STATUS_VALUES.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       TABLES
*"              TEXT STRUCTURE  TLINE
*"----------------------------------------------------------------------

  DATA: T_VALUES TYPE LEDID_T_STATUS,
        L_VALUE  TYPE LEDID_STATUS,
        L_LEN    TYPE I,
        L_MAX    TYPE I.

  CLEAR   TEXT.
  REFRESH TEXT.
* text-tdformat = '*'.

  CALL FUNCTION 'IDOC_STATUS_VALUES_READ'
       TABLES
            VALUES         = T_VALUES.

* get maximum length of field LAY_DESCRP
   LOOP AT T_VALUES INTO L_VALUE.
     L_LEN = STRLEN( L_VALUE-LAY_DESCRP ).
     IF L_LEN GT L_MAX.
       L_MAX = L_LEN.
     ENDIF.
   ENDLOOP.

   SORT T_VALUES BY VALUE.

* outgoing values
  TEXT = TEXT-190.    " status values for outgoing IDocs
  APPEND TEXT.
  LOOP AT T_VALUES INTO L_VALUE WHERE DIRECT EQ C_OUTGOING.
    PERFORM F07_FILL_STATUS_LINE USING TEXT
                                       L_VALUE
                                       L_MAX.
    APPEND TEXT.
  ENDLOOP.

* incoming values
  TEXT = TEXT-191.    " status values for outgoing IDocs
  APPEND TEXT.
  LOOP AT T_VALUES INTO L_VALUE WHERE DIRECT EQ C_INCOMING.
    PERFORM F07_FILL_STATUS_LINE USING TEXT
                                       L_VALUE
                                       L_MAX.
    APPEND TEXT.
  ENDLOOP.

ENDFUNCTION.
```

```
FUNCTION ZEDI_IDOC_PARSER.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(OBJECT) TYPE  LEDID_IDOCTYPE
*"             VALUE(OBJECT_TYPE) TYPE  LEDID_STRUCT_TYPE
*"             VALUE(OBJECT_RELEASE) LIKE  SY-SAPRL DEFAULT SY-SAPRL
*"             VALUE(VERSION) LIKE  EDI_VERREC-VERSION DEFAULT '3'
*"----------------------------------------------------------------------

DATA:  CIM_ATTRIB LIKE EDI_IAPI01.
RANGES: SEL_IDOCTYP FOR EDBAS-IDOCTYP.

CASE OBJECT_TYPE.
 WHEN LEDID_STRUCT_BASIS_IDOC.
    SEL_IDOCTYP-SIGN = 'I'.
    SEL_IDOCTYP-OPTION = 'EQ'.
    SEL_IDOCTYP-LOW = OBJECT.
    APPEND SEL_IDOCTYP.
    SUBMIT RSEIDOC3 AND RETURN
       WITH FLAG_REC EQ OFF
       WITH FLAG_ID EQ ON
       WITH IDOC_SEL IN SEL_IDOCTYP
       WITH REC_VRS EQ VERSION
       WITH IDOC_REL EQ OBJECT_RELEASE.
 WHEN LEDID_STRUCT_EXTENSION.
* read basis-idoc-type for extension
     CALL FUNCTION 'EXTTYPE_READ'
          EXPORTING
               PI_CIMTYP          =  OBJECT
          IMPORTING
               PE_ATTRIBUTES      =  CIM_ATTRIB
          EXCEPTIONS
               OTHERS       = 1.
     IF SY-SUBRC NE 0.
          MESSAGE ID SY-MSGID TYPE 'E'      NUMBER SY-MSGNO
                    WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
     ENDIF.
     IF CIM_ATTRIB-IDOCTYP EQ SPACE.
        MESSAGE I573 WITH OBJECT.
     ELSE.
      SUBMIT RSEIDOC3 AND RETURN
           WITH FLAG_REC EQ OFF
           WITH FLAG_ID EQ OFF
           WITH FLAG_EXT EQ ON
           WITH BAS_SEL EQ CIM_ATTRIB-IDOCTYP
           WITH EXT_SEL EQ OBJECT
           WITH REC_VRS EQ VERSION
           WITH IDOC_REL EQ OBJECT_RELEASE.
     ENDIF.
ENDCASE.

ENDFUNCTION.
```

```
FUNCTION ZEDI_IDOC_REC_PARSER.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(VERSION) LIKE  EDI_VERREC-VERSION DEFAULT '3'
*"             VALUE(FLAG_CONTROL_RECORD) TYPE  LEDID_FLAG
*"          DEFAULT 'X'
*"             VALUE(FLAG_DATA_RECORD) TYPE  LEDID_FLAG DEFAULT 'X'
*"             VALUE(FLAG_STATUS_RECORD) TYPE  LEDID_FLAG DEFAULT 'X'
*"----------------------------------------------------------------------

SUBMIT RSEIDOC3 AND RETURN
       WITH FLAG_REC EQ 'X'
       WITH FLAG_DC EQ FLAG_CONTROL_RECORD
       WITH FLAG_DD EQ FLAG_DATA_RECORD
       WITH FLAG_DS EQ FLAG_STATUS_RECORD
       WITH FLAG_ID EQ SPACE
       WITH REC_VRS EQ VERSION.
ENDFUNCTION.
```

```
FUNCTION ZEDI_IDOC_SHOW_TREE_LEGEND.
*"----------------------------------------------------------------------
*"*"Local interface:
*"----------------------------------------------------------------------

CALL SCREEN 200 STARTING AT 30 5 ENDING AT 70 20.

ENDFUNCTION.
```

```
FUNCTION ZEDI_SHOW_IDOC_AS_TREE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"            VALUE(ENABLE_FURTHER_OUTPUT) TYPE  LEDID_FLAG
*"         DEFAULT SPACE
*"            VALUE(OBJECT_RELEASE) LIKE  SY-SAPRL DEFAULT SY-SAPRL
*"            VALUE(PI_REC_VERSION) LIKE  EDI_VERREC-VERSION
*"         DEFAULT '3'
*"----------------------------------------------------------------------

* fill global variable G_OBJECT_RELEASE
  G_OBJECT_RELEASE = OBJECT_RELEASE.

* fill global variable G_OBJECT_VERSION
  G_OBJECT_VERSION = PI_REC_VERSION.

* an IDoc type is now displayed
  G_MODE = C_IDOC_TYPE.

* initialize global tables and data
   REFRESH T_EXCL.

  CALL FUNCTION 'EDI_CONVERT_IDOC_TAB_TO_TREE'
       TABLES
            TREE       = TREE.


  CALL FUNCTION 'RS_TREE_CONSTRUCT'
       TABLES
            NODETAB            = TREE.

IF ENABLE_FURTHER_OUTPUT EQ OFF.
  PERFORM FA2_DISABLE_FURTHER_OUTPUT.
ENDIF.

SET PF-STATUS 'IDOCTREE' EXCLUDING T_EXCL.

PERFORM F05_SET_TITLEBAR USING IDOC.

  CALL FUNCTION 'RS_TREE_LIST_DISPLAY'
       EXPORTING
            CALLBACK_PROGRAM          = 'SAPLEDIT'
            CALLBACK_USER_COMMAND     = 'FA1_HANDLE_TREE_COMMANDS'
            CALLBACK_MOREINFO_DISPLAY = 'FA1_HANDLE_MOREINFO'
            CALLBACK_COLOR_DISPLAY    = 'FA1_DISPLAY_COLOR_LEGEND'
            STATUS                    = 'OWN'.

ENDFUNCTION.
```

```
FUNCTION ZEDI_SHOW_IDOC_REC_AS_TREE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"            VALUE(ENABLE_FURTHER_OUTPUT) TYPE  LEDID_FLAG
*"          DEFAULT 'X'
*"            VALUE(VERSION) LIKE  EDI_VERREC-VERSION DEFAULT '2'
*"----------------------------------------------------------------------

* fill global variable G_OBJECT_VERSION
  G_OBJECT_VERSION = VERSION.

* IDoc records are now displayed
  G_MODE = C_IDOC_RECORD.

* initialize global tables
  REFRESH T_EXCL.

  CALL FUNCTION 'EDI_CONVERT_IDOC_REC_TO_TREE'
       EXPORTING
            VERSION           = VERSION
       TABLES
            TREE              = TREE.


  CALL FUNCTION 'RS_TREE_CONSTRUCT'
       TABLES
            NODETAB            = TREE.

IF ENABLE_FURTHER_OUTPUT EQ OFF.
  PERFORM FA2_DISABLE_FURTHER_OUTPUT.
ENDIF.

SET PF-STATUS 'IDOCTREE' EXCLUDING T_EXCL.

PERFORM F05_SET_TITLEBAR USING IDOC.

  CALL FUNCTION 'RS_TREE_LIST_DISPLAY'
       EXPORTING
            CALLBACK_PROGRAM          = 'SAPLEDIT'
            CALLBACK_USER_COMMAND     = 'FA1_HANDLE_TREE_COMMANDS'
            CALLBACK_MOREINFO_DISPLAY = 'FA1_HANDLE_MOREINFO'
            CALLBACK_COLOR_DISPLAY    = 'FA1_DISPLAY_COLOR_LEGEND'
              STATUS                    = 'OWN'.

ENDFUNCTION.
```

```
FUNCTION ZEDI_SHOW_INFO.
*"----------------------------------------------------------------------
*"*"Local interface:
*"----------------------------------------------------------------------

CALL SCREEN 300 STARTING AT 30 5 ENDING AT 60 12.

ENDFUNCTION.
```

```
FUNCTION ZEDI_SHOW_SEGMENT_AS_TREE.
*"----------------------------------------------------------------------
*""Local interface:
*"       IMPORTING
*"            VALUE(ENABLE_FURTHER_OUTPUT) TYPE  LEDID_FLAG
*"         DEFAULT SPACE
*"            VALUE(OBJECT_RELEASE) LIKE  SY-SAPRL DEFAULT SY-SAPRL
*"            VALUE(PI_REC_VERSION) LIKE  EDI_VERREC-VERSION
*"         DEFAULT '3'
*"----------------------------------------------------------------------
DATA: WA_IDOC TYPE LEDID_IDOC_STRUCT.

* fill global variable G_OBJECT_RELEASE
  G_OBJECT_RELEASE = OBJECT_RELEASE.

* fill global variable G_OBJECT_VERSION
  G_OBJECT_VERSION = PI_REC_VERSION.

* an IDoc type is now displayed
  G_MODE = C_SEGMENT.

* initialize global tables and data
   REFRESH T_EXCL.

CALL FUNCTION 'EDI_CONVERT_SEGMENT_TO_TREE'
     TABLES
          TREE    = TREE.

  CALL FUNCTION 'RS_TREE_CONSTRUCT'
       TABLES
            NODETAB            = TREE.

IF ENABLE_FURTHER_OUTPUT EQ OFF.
  PERFORM FA2_DISABLE_FURTHER_OUTPUT.
ENDIF.

  APPEND 'ZOUC' TO T_EXCL.
  APPEND 'ZOUI' TO T_EXCL.
  APPEND 'ZOUP' TO T_EXCL.
  SET PF-STATUS 'IDOCTREE' EXCLUDING T_EXCL.

LOOP AT T_IDOC INTO WA_IDOC.
* only 1 record in t_idoc
ENDLOOP.

PERFORM F05_SET_TITLEBAR_SEGMENT USING WA_IDOC-SEGMENT_TYPE.

  CALL FUNCTION 'RS_TREE_LIST_DISPLAY'
       EXPORTING
            CALLBACK_PROGRAM          = 'SAPLEDIT'
            CALLBACK_USER_COMMAND     = 'FA1_HANDLE_TREE_COMMANDS'
            CALLBACK_MOREINFO_DISPLAY = 'FA1_HANDLE_MOREINFO'
            CALLBACK_COLOR_DISPLAY    = 'FA1_DISPLAY_COLOR_LEGEND'
```

```
            STATUS                    = 'OWN'.

ENDFUNCTION.
```

```
FUNCTION ZEDI_START_IDOC_REC_TREE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(VERSION) LIKE  EDI_VERREC-VERSION DEFAULT '2'
*"             VALUE(FLAG_CONTROL_RECORD) TYPE  LEDID_FLAG
*"          DEFAULT 'X'
*"             VALUE(FLAG_DATA_RECORD) TYPE  LEDID_FLAG DEFAULT 'X'
*"             VALUE(FLAG_STATUS_RECORD) TYPE  LEDID_FLAG DEFAULT 'X'
*"             VALUE(ENABLE_FURTHER_OUTPUT) TYPE  LEDID_FLAG
*"          DEFAULT 'X'
*"----------------------------------------------------------------------


* initialize global tables
  REFRESH: T_IDOC_REC_STRUCT,
           T_IDOC_REC.

  G_CONTROL_FLAG = FLAG_CONTROL_RECORD.
  G_DATA_FLAG = FLAG_DATA_RECORD.
  G_STATUS_FLAG = FLAG_STATUS_RECORD.

  CALL FUNCTION 'IDOC_RECORD_EXT_STRUCTS_READ'
       EXPORTING
            VERSION             = VERSION
       TABLES
            IDOC_RECORDS_STRUCT = T_IDOC_REC_STRUCT
            IDOC_RECORDS        = T_IDOC_REC.

* deselect unwanted data
  IF FLAG_CONTROL_RECORD EQ OFF.
    DELETE T_IDOC_REC WHERE RECORD_TYPE EQ LEDID_CONTROL_RECORD_ID.
    DELETE T_IDOC_REC_STRUCT
                  WHERE RECORD_TYPE EQ LEDID_CONTROL_RECORD_ID.
  ENDIF.

  IF FLAG_DATA_RECORD EQ OFF.
    DELETE T_IDOC_REC WHERE RECORD_TYPE EQ LEDID_DATA_RECORD_ID.
    DELETE T_IDOC_REC_STRUCT
                  WHERE RECORD_TYPE EQ LEDID_DATA_RECORD_ID.
  ENDIF.

  IF FLAG_STATUS_RECORD EQ OFF.
    DELETE T_IDOC_REC WHERE RECORD_TYPE EQ LEDID_STATUS_RECORD_ID.
    DELETE T_IDOC_REC_STRUCT
                  WHERE RECORD_TYPE EQ LEDID_STATUS_RECORD_ID.
  ENDIF.

  CALL FUNCTION 'EDI_SHOW_IDOC_REC_AS_TREE'
       EXPORTING
            ENABLE_FURTHER_OUTPUT = ENABLE_FURTHER_OUTPUT
            VERSION               = VERSION.
```

```
ENDFUNCTION.
```

```
FUNCTION ZEDI_START_IDOC_TREE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(OBJECT) TYPE  LEDID_IDOCTYPE
*"             VALUE(OBJECT_TYPE) TYPE  LEDID_STRUCT_TYPE
*"             VALUE(OBJECT_RELEASE) LIKE  SY-SAPRL DEFAULT SY-SAPRL
*"             VALUE(PI_REC_VERSION) LIKE  EDI_VERREC-VERSION
*"         DEFAULT '3'
*"             VALUE(ENABLE_FURTHER_OUTPUT) TYPE  LEDID_FLAG
*"         DEFAULT 'X'
*"----------------------------------------------------------------------


* initialize global tables
  REFRESH: T_IDOC,
           T_SEGMENTS,
           T_FIELDS.

  CALL FUNCTION 'IDOC_TYPE_COMPLETE_READ'
       EXPORTING
            STRUCT_TYPE    = OBJECT_TYPE
            IDOCTYPE       = OBJECT
            RELEASE        = OBJECT_RELEASE
            VERSION        = PI_REC_VERSION
       IMPORTING
            IDOC_TYPE      = IDOC
       TABLES
            IDOC_STRUCT    = T_IDOC
            SEGMENTS       = T_SEGMENTS
            SEGMENT_STRUCT = T_FIELDS
       EXCEPTIONS
            OTHERS         = 1.
  CASE SY-SUBRC.
    WHEN 0.
      CALL FUNCTION 'EDI_SHOW_IDOC_AS_TREE'
           EXPORTING
                ENABLE_FURTHER_OUTPUT = ENABLE_FURTHER_OUTPUT
                OBJECT_RELEASE        = OBJECT_RELEASE
                PI_REC_VERSION        = PI_REC_VERSION.
    WHEN OTHERS.
      MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
                 WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDCASE.
ENDFUNCTION.
```

```
FUNCTION ZEDI_START_SEGMENT_TREE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(SEGMENT_TYPE) LIKE  EDI_IAPI02-SEGTYP
*"             VALUE(OBJECT_RELEASE) LIKE  SY-SAPRL DEFAULT SY-SAPRL
*"             VALUE(PI_REC_VERSION) LIKE  EDI_VERREC-VERSION
*"          DEFAULT '3'
*"             VALUE(ENABLE_FURTHER_OUTPUT) TYPE  LEDID_FLAG
*"          DEFAULT 'X'
*"----------------------------------------------------------------------

* local data
 DATA: WA_IDOC TYPE LEDID_IDOC_STRUCT,
       S_HEADER LIKE EDISEGMHD.
* initialize global tables
  REFRESH: T_IDOC,
           T_SEGMENTS,
           T_FIELDS.
*global data
 SEGMENT = SEGMENT_TYPE.

CALL FUNCTION 'IDOC_COMPLETE_SEGMENT_READ'
     EXPORTING
          SEGTYP                     = SEGMENT_TYPE
          RELEASE                    = OBJECT_RELEASE
          VERSION                    = PI_REC_VERSION
     TABLES
          SEGMENT                    = T_SEGMENTS
          SEGMENT_STRUCT             = T_FIELDS
     EXCEPTIONS
          SEGMENT_UNKNOWN            = 1
          SEGMENT_STRUCTURE_UNKNOWN = 2
          SEGMENT_NOT_IN_RELEASE     = 3
          SEGMENT_INCONSISTENT       = 4
          OTHERS                     = 5.
IF SY-SUBRC NE 0.
    MESSAGE ID SY-MSGID TYPE 'E' NUMBER SY-MSGNO WITH
        SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.

* read the segment-info

CALL FUNCTION 'SEGMENT_READ'
     EXPORTING
          SEGMENTTYP           = SEGMENT_TYPE
    IMPORTING
         SEGMENTHEADER        = S_HEADER
     EXCEPTIONS
          NO_AUTHORITY         = 1
          SEGMENT_NOT_EXISTING = 2
          OTHERS               = 3.
```


```
IF SY-SUBRC NE 0.
    MESSAGE ID SY-MSGID TYPE 'E' NUMBER SY-MSGNO WITH
        SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.


* fill t_idoc

          WA_IDOC-SEGMENT_TYPE = SEGMENT_TYPE.
          WA_IDOC-SEGMENT_TYPE_ATTRIB-DESCRP = S_HEADER-DESCRP.
          APPEND WA_IDOC TO T_IDOC.

CALL FUNCTION 'EDI_SHOW_SEGMENT_AS_TREE'
     EXPORTING
         ENABLE_FURTHER_OUTPUT = ENABLE_FURTHER_OUTPUT
         OBJECT_RELEASE        = OBJECT_RELEASE
         PI_REC_VERSION        = PI_REC_VERSION
     EXCEPTIONS
          OTHERS                = 1.

IF SY-SUBRC NE 0.
    MESSAGE ID SY-MSGID TYPE 'E' NUMBER SY-MSGNO WITH
        SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.


ENDFUNCTION.
```

```
FUNCTION ZEDI_START_SYIDOC01.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(OBJECT) TYPE  LEDID_IDOCTYPE
*"             VALUE(OBJECT_TYPE) TYPE  LEDID_STRUCT_TYPE
*"             VALUE(OBJECT_RELEASE) LIKE  SY-SAPRL DEFAULT SY-SAPRL
*"             VALUE(PI_REC_VERSION) LIKE  EDI_VERREC-VERSION
*"         DEFAULT '3'
*"----------------------------------------------------------------------

* initialize global tables
  REFRESH: T_IDOC,
           T_SEGMENTS,
           T_FIELDS.

* get IDoc structure
  CALL FUNCTION 'IDOC_TYPE_COMPLETE_READ'
       EXPORTING
            STRUCT_TYPE    = OBJECT_TYPE
            IDOCTYPE       = OBJECT
            RELEASE        = OBJECT_RELEASE
            VERSION        = PI_REC_VERSION
       IMPORTING
            IDOC_TYPE      = IDOC
       TABLES
            IDOC_STRUCT    = T_IDOC
            SEGMENTS       = T_SEGMENTS
            SEGMENT_STRUCT = T_FIELDS
       EXCEPTIONS
            OTHERS         = 1.
  CASE SY-SUBRC.
    WHEN 0.
* read current configuration
      PERFORM FA2_GET_USER_ATTRIBUTES_ALL.

* set attributes
      PERFORM FA3_SET_ATTRIBUTES USING EDI_ATTRIB.

* create an SYIDOC01-IDoc
      CALL FUNCTION 'EDI_CREATE_SYIDOC01'
           EXPORTING
                OBJECT_RELEASE = OBJECT_RELEASE.
    WHEN OTHERS.
      MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
                 WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDCASE.
ENDFUNCTION.
```

```
FUNCTION ZEDI_START_SYRECD01.
*"----------------------------------------------------------------------
*"*"Local interface:
*"       IMPORTING
*"             VALUE(VERSION) LIKE  EDI_VERREC-VERSION DEFAULT '2'
*"             VALUE(FLAG_CONTROL_RECORD) TYPE  LEDID_FLAG
*"         DEFAULT 'X'
*"             VALUE(FLAG_DATA_RECORD) TYPE  LEDID_FLAG DEFAULT 'X'
*"             VALUE(FLAG_STATUS_RECORD) TYPE  LEDID_FLAG DEFAULT 'X'
*"----------------------------------------------------------------------


* initialize global tables
  REFRESH: T_IDOC_REC,
           T_IDOC_REC_STRUCT.

* get structures of control/data/status records
  CALL FUNCTION 'IDOC_RECORD_EXT_STRUCTS_READ'
       EXPORTING
            VERSION                      = VERSION
       TABLES
            IDOC_RECORDS_STRUCT          = T_IDOC_REC_STRUCT
            IDOC_RECORDS                 = T_IDOC_REC.

* deselect unwanted data
  IF FLAG_CONTROL_RECORD EQ OFF.
    DELETE T_IDOC_REC WHERE RECORD_TYPE EQ LEDID_CONTROL_RECORD_ID.
    DELETE T_IDOC_REC_STRUCT
                  WHERE RECORD_TYPE EQ LEDID_CONTROL_RECORD_ID.
  ENDIF.

  IF FLAG_DATA_RECORD EQ OFF.
    DELETE T_IDOC_REC WHERE RECORD_TYPE EQ LEDID_DATA_RECORD_ID.
    DELETE T_IDOC_REC_STRUCT
                  WHERE RECORD_TYPE EQ LEDID_DATA_RECORD_ID.
  ENDIF.

  IF FLAG_STATUS_RECORD EQ OFF.
    DELETE T_IDOC_REC WHERE RECORD_TYPE EQ LEDID_STATUS_RECORD_ID.
    DELETE T_IDOC_REC_STRUCT
                  WHERE RECORD_TYPE EQ LEDID_STATUS_RECORD_ID.
  ENDIF.

  PERFORM FA2_GET_USER_ATTRIBUTES_ALL.
  PERFORM FA3_SET_ATTRIBUTES USING EDI_ATTRIB.

* create an SYRECD01-IDoc
  CALL FUNCTION 'EDI_CREATE_SYRECD01'
       EXPORTING
            VERSION                = VERSION.

ENDFUNCTION.
```

```
FUNCTION zget_pathname_filename.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     REFERENCE(P_DEFAULT_FILE) LIKE  EDI_IAPI00-IDOCTYP OPTIONAL
*"  CHANGING
*"     REFERENCE(P_PATH) LIKE  RLGRAP-FILENAME
*"     REFERENCE(P_FILE) LIKE  RLGRAP-FILENAME
*"     REFERENCE(P_BROWSER) LIKE  EDI_HTMFIL-EDI_START
*"     REFERENCE(P_RC) LIKE  SY-SUBRC
*"     REFERENCE(P_FULLNAME) TYPE  STRING
*"----------------------------------------------------------------------

  DATA :   l_settings   LIKE edi_htmfil,
           l_where      LIKE edipar-edi_paorig.

** first we try to read a user specific setting
  CALL FUNCTION 'IDOC_CONFIGURE_DOC'
       EXPORTING
            service            = 'R'
       IMPORTING
            parameter_origin   = l_where
       CHANGING
            htmlfile_data      = l_settings
       EXCEPTIONS
            undefined_service  = 1
            user_action_cancel = 2
            OTHERS             = 3.

  IF sy-subrc EQ 0.
    IF l_where EQ space.               "read default path
** what is our actual working directory
      CALL FUNCTION 'WS_QUERY'
           EXPORTING
                query          = 'CD'
           IMPORTING
                return         = l_settings-edi_pthnht
           EXCEPTIONS
                inv_query      = 1
                no_batch       = 2
                frontend_error = 3
                OTHERS         = 4.
      IF sy-subrc NE 0.
        MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH
                   sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
      ENDIF.
    ENDIF.

    p_path    = l_settings-edi_pthnht.
    p_file    = l_settings-edi_filhtm.
    p_browser = l_settings-edi_start.
    IF p_file EQ space.
      p_file = p_default_file.
```

```
    ENDIF.
    CALL FUNCTION 'IDOC_WS_DISPLAY'
         EXPORTING
              title_popup      = text-610
         CHANGING
              def_filename     = p_file
              def_path         = p_path
         EXCEPTIONS
              selection_cancel = 1
              selection_error  = 2
              OTHERS           = 3.
    CASE sy-subrc.
      WHEN 0.
        p_rc = 0.
      WHEN 1.
        MESSAGE i222.
        p_rc = 4.
      WHEN 2.
* environment error; use p_file and p_path with default values
        p_rc = 0.
      WHEN OTHERS.
        MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
                WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDCASE.
  ELSEIF sy-subrc EQ 2.
    p_rc = 4.
    MESSAGE i222.                      " user canceled setting
  ELSE.
    p_rc = 4.
  ENDIF.

  DATA: l_line(180) TYPE c,
        l_answer    TYPE c.

  IF p_file EQ space.
    p_rc = 4.
  ELSE.
    CONCATENATE p_path p_file INTO p_fullname.
  ENDIF.

ENDFUNCTION.
```

```
FUNCTION-POOL ZEDIT       MESSAGE-ID E0   NO STANDARD PAGE HEADING.

INCLUDE <SYMBOL>.

TYPE-POOLS: LEDID.

TABLES: EDI_ATTRIB,
        EDI_HTMFIL,
        EDI_IDCATT,
        EDI_CFILE,
        T000,
        EDIMSG,
        EDIMSGT,
        E1IDOCH,
        E1IDOCT,
*       e1txth1,
*       e1txtp1,
        E1TXTH2,
        E1TXTP2,
        E1TXTH3,
        E1TXTP3,
        E1SEGMH,
        E1SEGMF,
*       e1valu1,
        E1MESH.


*-global data---------------------------------------------------------------
  DATA: IDOC                TYPE LEDID_IDOC_TYPE,
        SEGMENT             LIKE EDI_IAPI02-SEGTYP,
        T_IDOC              TYPE LEDID_T_IDOC_STRUCT,
        T_SEGMENTS          TYPE LEDID_T_SEGMENT,
        T_FIELDS            TYPE LEDID_T_SEGMENT_STRUCT,
        T_IDOC_REC          TYPE LEDID_T_IDOC_RECORD,
        T_IDOC_REC_STRUCT   TYPE LEDID_T_IDOC_RECORD_STRUCT,
        T_EXCL              LIKE SY-TCODE OCCURS 1 WITH HEADER LINE,
        TREE                LIKE SNODETEXT OCCURS 1 WITH HEADER LINE,
        G_OBJECT_RELEASE    LIKE SY-SAPRL,
        G_OBJECT_VERSION    LIKE EDI_VERREC-VERSION,
        G_CONTROL_FLAG       TYPE LEDID_FLAG,
        G_DATA_FLAG         TYPE LEDID_FLAG,
        G_STATUS_FLAG       TYPE LEDID_FLAG,
        BEGIN OF G_SEGNUM OCCURS 10,
          SEGNAM LIKE EDIDD-SEGNAM,
          SEGNUM LIKE EDIDD-SEGNUM,
        END OF G_SEGNUM.
*---------------------------------------------------------------------------

DATA: LEV_OFFSET        TYPE I,
      G_MODE            TYPE C,      " mode: record or IDoc type display
      OK_CODE           LIKE SY-UCOMM,
      OK_CODE_INFO      LIKE SY-UCOMM,
      INT_LAYOUT      LIKE SEUTEXPAND   OCCURS 20 WITH HEADER LINE,
```

```
      LAYOUT_CCOLUMN  LIKE SY-CUCOL,
      LAYOUT_CLINE    LIKE SY-CUROW,
      LAYOUT_LCOLUMN  LIKE SY-STACO,
      LAYOUT_LLINE    LIKE SY-STARO.

CONSTANTS:
       ON(1)           TYPE C               VALUE 'X',
       OFF(1)          TYPE C               VALUE ' ',
       DONT_DISPLAY(1) TYPE C               VALUE '/',
       ICON_LOCKED(20) TYPE C               VALUE 'ICON_LED_GREEN',
       ICON_UNLOCKED(20) TYPE C             VALUE 'ICON_LED_YELLOW',
       ICON_ISOCODE(20) TYPE C              VALUE 'ICON_ISO_CODE',
       SYM_ATTR_YES(20) TYPE C              VALUE 'SYM_FILLED_SQUARE',
       SYM_ATTR_NO(20) TYPE C               VALUE 'SYM_SQUARE',
       SYM_DOCU_YES(20) TYPE C              VALUE 'SYM_FILLED_CIRCLE',
       SYM_DOCU_NO(20) TYPE C               VALUE 'SYM_CIRCLE',
       SYM_VALU_YES(20) TYPE C              VALUE 'SYM_FILLED_DIAMOND',
       SYM_VALU_NO(20) TYPE C               VALUE 'SYM_DIAMOND',
       TYPE_HEADER                          VALUE 'H',
       TYPE_REC_HEADER(4)                   VALUE 'RH',
       TYPE_SEGMENT(4)                      VALUE 'TB',
       TYPE_REC_NODE(4)                     VALUE 'RN',
       TYPE_FIELDNODE                       VALUE 'D',
       TYPE_FIELD(4)                        VALUE 'DE',
       TYPE_REC_FIELD(4)                    VALUE 'RF',
       LENGTH_TEXT     TYPE I               VALUE 70,
       LENGTH_SEGMENT  TYPE I               VALUE 30,
       LENGTH_FIELD    TYPE I               VALUE 30,
       LENGTH_ICON     TYPE I               VALUE 2,
       LENGTH_SYMBOL   TYPE I               VALUE 1,
       COL_HEADER      TYPE I               VALUE 1,
       COL_SEGMENT     TYPE I               VALUE 4,
       COL_TEXT        TYPE I               VALUE 0,
       COL_DOCU        TYPE I               VALUE 2,
       COL_ATTR        TYPE I               VALUE 1,
       COL_CHECKBOX    TYPE I               VALUE 3,
       COL_FIELDNODE   TYPE I               VALUE 2,
       INTENS_HEADER                        VALUE '1',
       INTENS_SEGMENT_NODE                  VALUE '1',
       INTENS_SEGMENT_LEAF                  VALUE '0',
       INTENS_TEXT                          VALUE '0',
       INTENS_DOCU                          VALUE '1',
       INTENS_ATTR                          VALUE '0',
       INTENS_FIELDNODE                     VALUE '0',
       INTENS_CHECKBOX                      VALUE '0',
       KIND_ICON                            VALUE 'I',
       KIND_SYMBOL                          VALUE 'S',
       C_COLON                              VALUE ':',
       C_OUTGOING                           VALUE '1',
       C_INCOMING                           VALUE '2',
       C_PARAMETERS_READ                    VALUE 'R',
       C_PARAMETERS_SET                     VALUE 'T',
       C_MESTYP_SYIDOC         LIKE EDP13-MESTYP VALUE 'SYIDOC',
```

```
       C_MESTYP_SYRECD         LIKE EDP13-MESTYP VALUE 'SYRECD',
       C_LOGICAL_SYSTEM        LIKE EDP13-RCVPRT VALUE 'LS',
       C_FIELD_TEXT(20)        TYPE C        VALUE 'P_TREE-TEXT',
       C_FIELD_TLENGTH(20)     TYPE C        VALUE 'P_TREE-TLENGTH',
       C_FIELD_TCOLOR(20)      TYPE C        VALUE 'P_TREE-TCOLOR',
       C_FIELD_TINTENSIV(20)   TYPE C        VALUE 'P_TREE-TINTENSIV',
       C_TEXT                                VALUE 'T',
       C_DOCU                                VALUE 'D',
       C_ATTR                                VALUE 'A',
       C_IDOC_TYPE                           VALUE 'I',
       C_IDOC_RECORD                         VALUE 'R',
       C_SEGMENT                             VALUE 'S',
       C_EDI_STATUS(10)        TYPE C        VALUE 'EDI_STATUS',
       C_EDI4STATUS(10)        TYPE C        VALUE 'EDI4STATUS',
       C_EDI_DC(10)            TYPE C        VALUE 'EDI_DC    ',
       C_EDI_DC40(10)          TYPE C        VALUE 'EDI_DC40  ',
       C_EDI_DD(10)            TYPE C        VALUE 'EDI_DD    ',
       C_EDI_DD40(10)          TYPE C        VALUE 'EDI_DD40  ',
       C_EDI_DS(10)            TYPE C        VALUE 'EDI_DS    ',
       C_EDI_DS30(10)          TYPE C        VALUE 'EDI_DS30  ',
       C_EDI_DS40(10)          TYPE C        VALUE 'EDI_DS40  ',
       C_TABL                  TYPE LEDID_OBJTYPE VALUE 'TB',
       C_DTEL                  TYPE LEDID_OBJTYPE VALUE 'DE',
       C_READY_FOR_DISPATCHING LIKE EDIDS-STATUS VALUE '30'.

DEFINE M_SET_ATTR_DISPLAY.
  &1 = COL_ATTR.
  &2 = INTENS_ATTR.
END-OF-DEFINITION.

DEFINE M_SET_TEXT_DISPLAY.
  &1 = COL_TEXT.
  &2 = INTENS_TEXT.
END-OF-DEFINITION.

DEFINE M_SET_DOCU_DISPLAY.
  &1 = COL_DOCU.
  &2 = INTENS_DOCU.
END-OF-DEFINITION.

DEFINE M_TOGGLE.       " switch between ON and OFF
  CASE &1.
    WHEN ON.
      &1 = OFF.
    WHEN OFF.
      &1 = ON.
  ENDCASE.
END-OF-DEFINITION.
```

Author Bio

Yukai (Steven) Shi has been with SAP Labs since 2003. He has worked on IndustrySpeak projects where he has been actively involved in development of Business Packages for HighTech (RosettaNet), Chemical (CIDX), as well as Automotive (STAR) standards. He has been extensively working on mapping solutions using Exchange Infrastructure (XI) graphical mapping tool.