S4HANA to Facele - Document Compliance

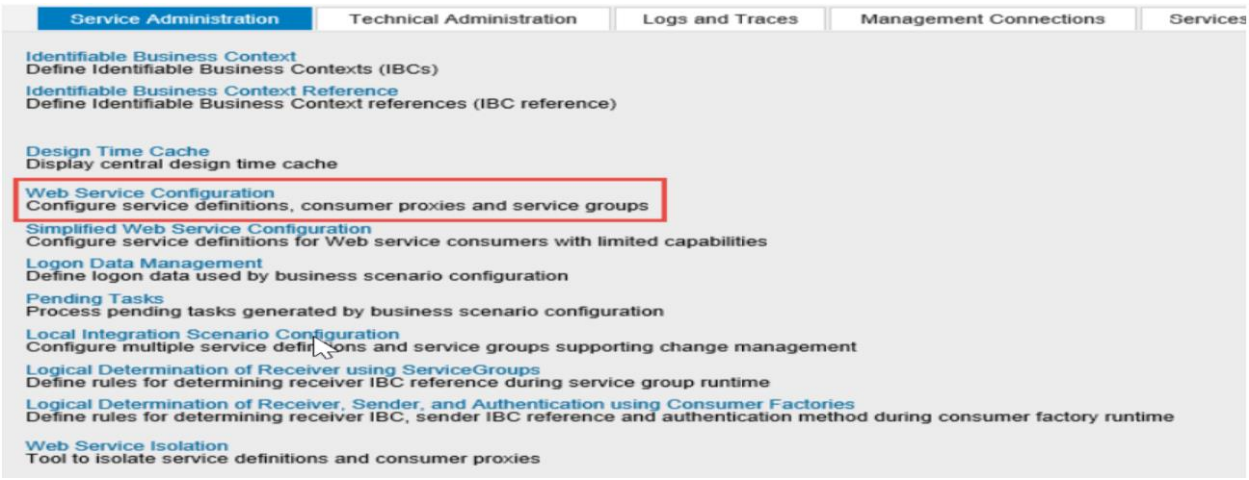
Table of Contents

Purpose

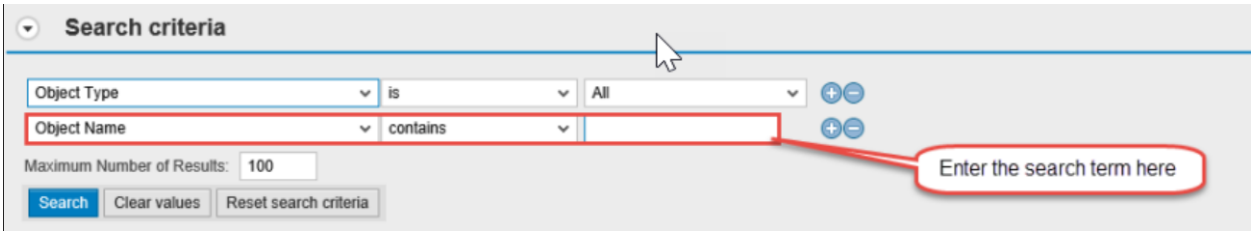
Quick start to development of Document Compliance: Electronic Documents for Chile using Service provider Facele.

SOAMANAGER Configuration

SOAMANAGER transaction and search for Web Service.

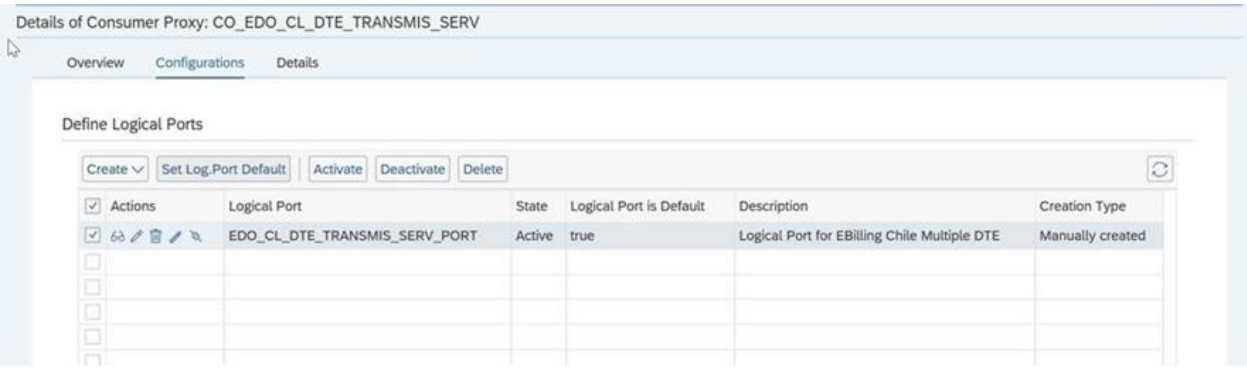


Find the proxies for Chile with search term CO_EDO_CL*.



Configure the Logical port for the Proxy below:

Proxy Name	Logical Port Name	Endpoint url
CO_EDO_CL_DTE_TRANSMIS_SERV	EDO_CL_DTE_TRANS-MIS_SERV_PORT	/cxf/ChileSendMultipleDTE

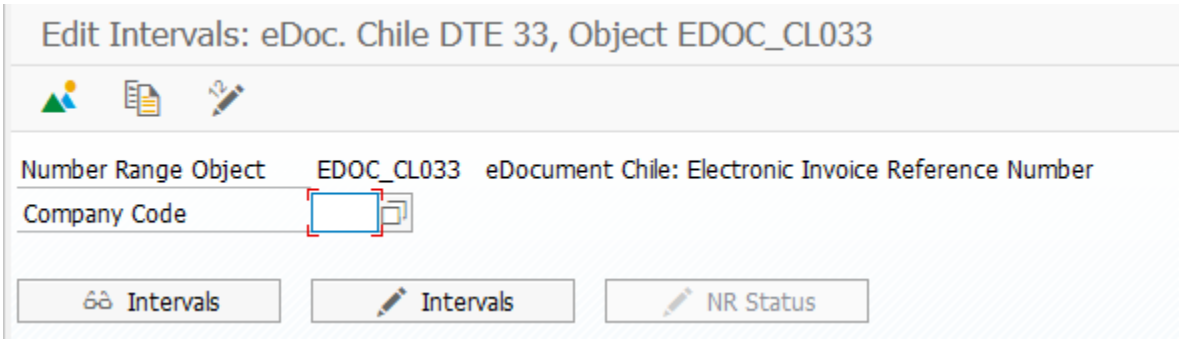


S/4 HANA Configuration

Step 1: Number range

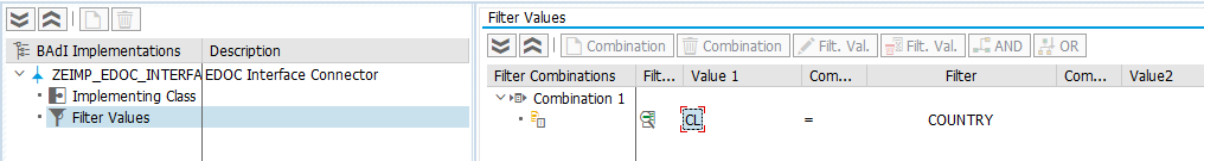
Maintain a number range starting EDOC_CL + DTE Type

e.g. EDOC_CL033



Step 2: BADI Implementation

1. **EDOC_INTERFACE_CONNECTOR**



Please refer to: Appendix 1 ->

BADI Implementation

2. **Implement Function Modules**
   The trigger method calls Function Module:

   Function Module 1

Please refer to: Appendix 2 -> Appendix - 2

Function Module 1

Function Module 2

Please refer to: Appendix 2 -> Function Module 2

Below attached code is used in FUNC_2 to Reverts the number range in case of any failure.
Please refer to: Appendix 3 ->

Reverts the number range after failure case

Note: To not get the PDF remove code between <<******** PDF related **********>> tags in FUNC_2**

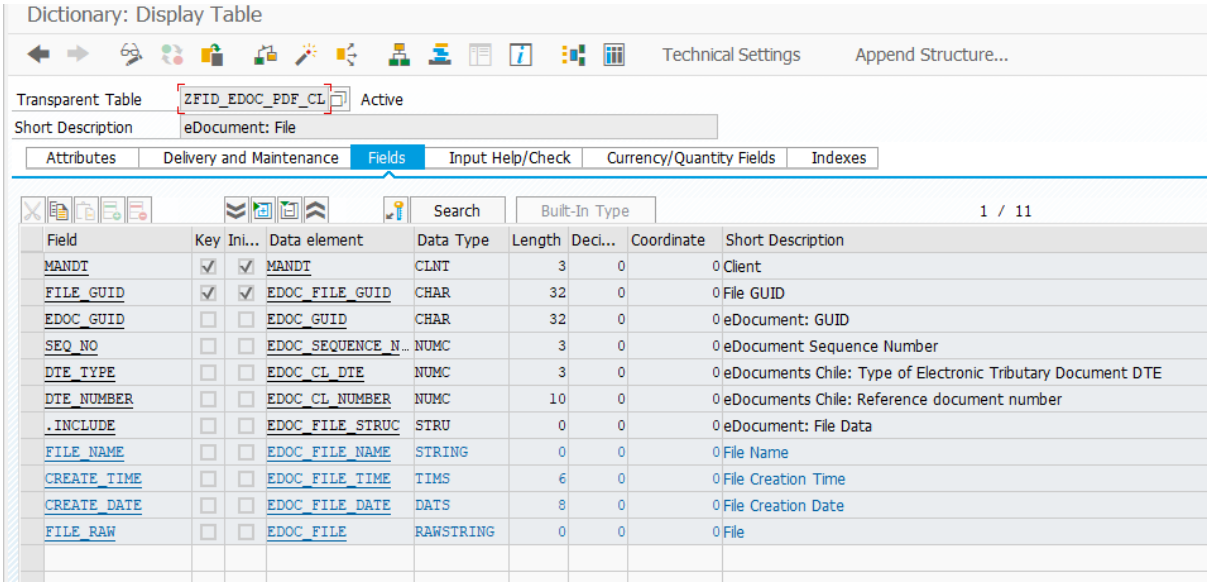
Step 3: Status Update Post Exit

Finally for status update write post-exit on CL_EDOC_PROCESS method UPDATE_EDOCUMENT

Please refer to: Appendix 3 -> POST- Exit method to update E-document

Step 4: Handling PDF response

1. **Create a table:**



Field	Key	Ini...	Data element	Data Type	Length	Deci...	Coordinate	Short Description
MANDT	☑	☑	MANDT	CLNT	3	0	0	Client
FILE_GUID	☑	☑	EDOC_FILE_GUID	CHAR	32	0	0	File GUID
EDOC_GUID	☐	☐	EDOC_GUID	CHAR	32	0	0	eDocument: GUID
SEQ_NO	☐	☐	EDOC_SEQUENCE_N...	NUMC	3	0	0	eDocument Sequence Number
DTE_TYPE	☐	☐	EDOC_CL_DTE	NUMC	3	0	0	eDocuments Chile: Type of Electronic Tributary Document DTE
DTE_NUMBER	☐	☐	EDOC_CL_NUMBER	NUMC	10	0	0	eDocuments Chile: Reference document number
.INCLUDE	☐	☐	EDOC_FILE_STRUC	STRU	0	0	0	eDocument: File Data
FILE_NAME	☐	☐	EDOC_FILE_NAME	STRING	0	0	0	File Name
CREATE_TIME	☐	☐	EDOC_FILE_TIME	TIMS	6	0	0	File Creation Time
CREATE_DATE	☐	☐	EDOC_FILE_DATE	DATS	8	0	0	File Creation Date
FILE_RAW	☐	☐	EDOC_FILE	RAWSTRING	0	0	0	File

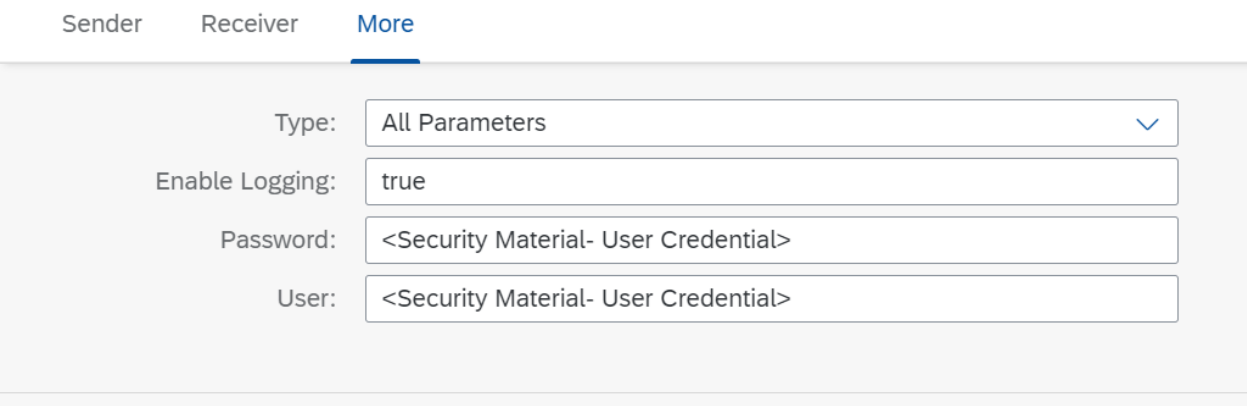
2. **Create an FM:**
   Please refer to: Appendix 3 -> Function Module for PDF response

3. **Create an override exit on method display_pdf of class cl_edoc_action_ui.**
   Please refer to: Appendix 3 -> Display PDF response

Integration Specification

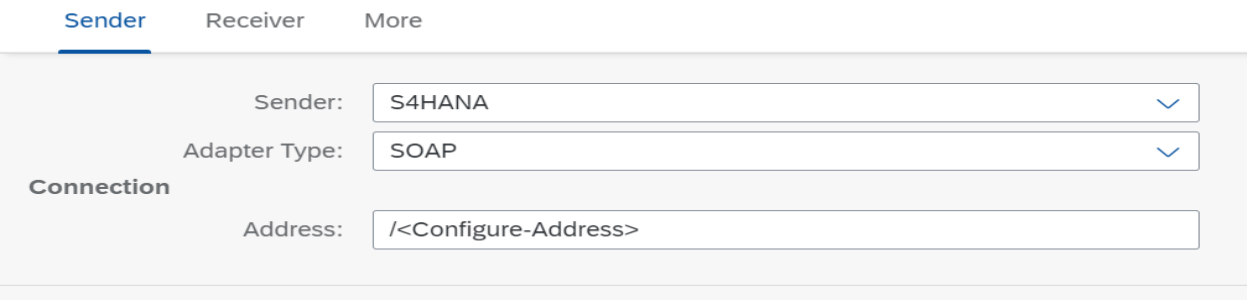
Type of Integration (outbound or inbound)	Outbound
Data Source Entities	S4HANA
Average Number of Records	1
Maximum number of Records	1
Processing Type	Request-Reply
Job Schedule	Flexible - As per business Requirement
Data Description (xml, comma delimited etc…)	Xml
Target Location	Facele

Parameters Usage Guide



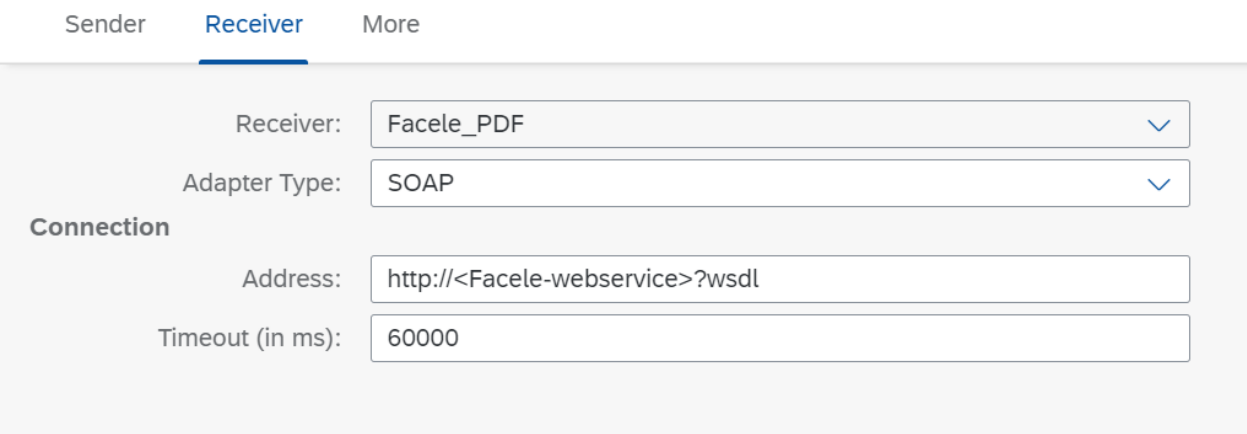
Endpoint Configuration

Soap Sender



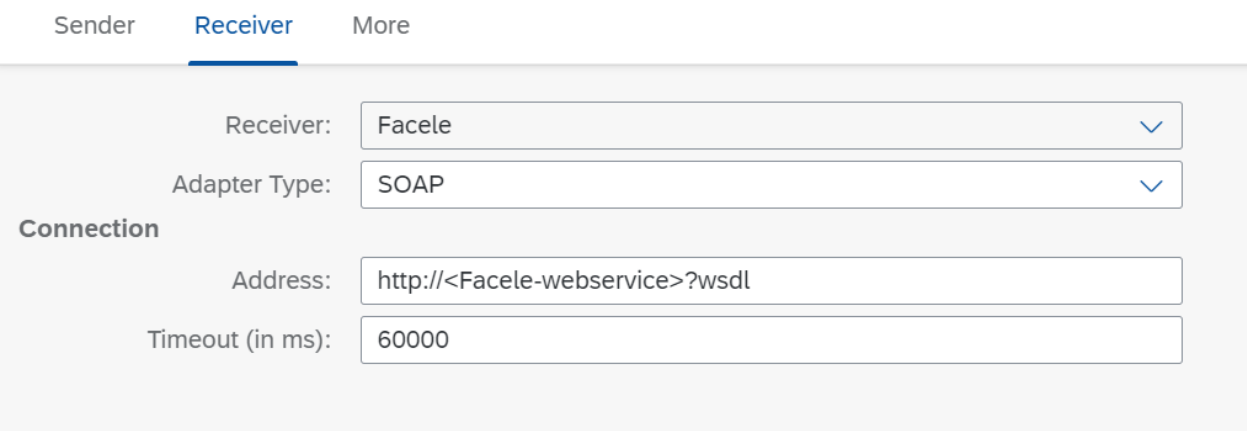
Soap Receiver (PDF Response)

Note: Import the server certificate in CPI tenant before configuring the channel.



Soap Receiver (Xml Response)

Note: Import the server certificate in CPI tenant before configuring the channel.



Appendix - 1

BADI Implementation

```
METHOD if_edoc_interface_connector~trigger.
*    include EDOC_INTF_CONN_TRIGGER.
* <SIGNATURE>---------------------------------------------------------------------------------------+
* | Instance Public Method IF_EDOC_INTERFACE_CONNECTOR~TRIGGER
* +--------------------------------------------------------------------------------------------------+
* | [--->] IV_EDOC_GUID               TYPE       EDOC_GUID
* | [--->] IO_EDOCUMENT                 TYPE REF TO CL_EDOCUMENT
* | [--->] IV_TEST_MODE               TYPE       SAP_BOOL (default =SPACE)
* | [--->] IV_INTERFACE_ID            TYPE       EDOC_INTERFACE_ID(optional)
* | [--->] IV_INT_VERSION             TYPE       EDOC_INT_VERSION(optional)
* | [--->] IO_INTF_CONN_IMPL          TYPE REF TO IF_EDOC_INTF_CONN_IMPL(optional)
* | [!CX!] CX_EDOCUMENT
* +--------------------------------------------------------------------------------------</SIGNATURE>
*  METHOD if_edoc_interface_connector~trigger.


  DATA: BEGIN OF ls_interface,
        ns       TYPE /aif/ns,
        ifname    TYPE /aif/ifname,
        ifversion TYPE /aif/ifversion,
       END OF ls_interface.


  DATA: lv_msgid        TYPE sxmsmguid,
      ls_edoint       TYPE edoint,
      lv_interface_id  TYPE edoc_interface_id,
      lv_ns           TYPE string,
      lv_ifname       TYPE string,
      lv_ifversion     TYPE string,
      lv_log_port_name TYPE prx_logical_port_name,
      lv_error_txt     TYPE string.


  DATA: lo_edoc_config_db TYPE REF TO cl_edoc_config_db,
      lo_edoc_util      TYPE REF TO cl_edoc_util,
```

```
    ld_source_data    TYPE REF TO data,
    lo_class_desc     TYPE REF TO cl_abap_typedescr,
    lv_class_name     TYPE string.

DATA: lv_process_step TYPE edoc_process_step,
    lv_subrc       TYPE sysubrc,
    lx_root        TYPE REF TO cx_root.

DATA: ld_aif_data     TYPE REF TO data,
    lv_if_guid      TYPE edoc_interface_guid,
    ld_process_data TYPE REF TO data.

DATA: lo_edocument    TYPE REF TO cl_edocument.
DATA: lv_comm_action  TYPE abap_bool.

DATA: lv_proxy         TYPE /aif/proxy_outbound,
    lv_target_struc   TYPE /aif/if_ddic_struct,
    ld_target_struc   TYPE REF TO data,
    lo_intf_conn_impl TYPE REF TO if_edoc_intf_conn_impl.

DATA: lo_edoc_dpp TYPE REF TO cl_edoc_dpp,            "2587421
    lt_edobupa  TYPE edobupa_tab.                  "2587421

DATA: lv_method TYPE seocpdname,                    "2908049
    lv_class  TYPE seoclsname,                    "2908049
    ls_tmp    TYPE edo_cl_envio_dte1.

CONSTANTS: lc_cl_map_cl       TYPE edoc_class_name VALUE 'CL_EDOC_MAP_CL', "2908049
        lc_cl_map_edocument TYPE edoc_class_name VALUE 'CL_EDOC_MAP_CL_EDOCUMENT', "2908049
        lc_method          TYPE seocpdname VALUE 'GET_SOURCE_DOCUMENT_DATA'. "2908049

FIELD-SYMBOLS: <ls_source_structure> TYPE any,
           <ls_target_structure> TYPE any.

" Get eDocument
```

```
  cl_edocument=>retrieve_by_edoc_guid( EXPORTING iv_edoc_guid = iv_edoc_guid
                                RECEIVING ro_edocument = lo_edocument ).



  " Get data from source document
  IF lo_edocument->ms_edocument-edoc_type EQ 'CL_RTF'.    "2908049
    lv_class = lc_cl_map_edocument.
  ELSE.
    lv_class = lc_cl_map_cl.
  ENDIF.


  lv_method = lc_method.
  TRY.
      CALL METHOD (lv_class)=>(lv_method)
        EXPORTING
          io_edocument   = lo_edocument
        RECEIVING
          rd_source_data = ld_source_data.
    CATCH cx_dynamic_check.
      MESSAGE e083(edocument) WITH lv_class lv_method INTO cl_edocument=>gv_error_txt.
      cl_edocument=>raise_edoc_exception( ).
  ENDTRY.
*   Get configuration DB handler. Allows using stub with Unit Test
  CREATE OBJECT lo_edoc_config_db TYPE cl_edoc_config_db.


*   Get interface connector implementation
  IF io_intf_conn_impl IS BOUND.
    lo_intf_conn_impl = io_intf_conn_impl.
  ELSE.
    lo_intf_conn_impl = me. "S4CORE only
  ENDIF.
*
**-- Prepare structure with source data
*    ld_source_data = lo_intf_conn_impl->get_source_data(
*                  io_edocument = io_edocument ).
```

```
  ASSIGN ld_source_data->* TO <ls_source_structure>.

*   ls_tmp = <ls_source_structure>.

  IF <ls_source_structure> IS NOT ASSIGNED.
    lo_class_desc = cl_abap_classdescr=>describe_by_object_ref( me ).
    lv_class_name = lo_class_desc->get_relative_name( ).
    MESSAGE e125(edocument)
      WITH 'ld_source_data->*' '<ls_source_structure>'
           lv_class_name 'TRIGGER'
      INTO cl_edocument=>gv_error_txt.
    cl_edocument=>raise_edoc_exception( ).
  ENDIF.

*-- Logical port and interface determination
  lv_interface_id = iv_interface_id.
  IF iv_interface_id IS INITIAL.
    lv_process_step = 'SENDEDOC'.
*   Select the interface ID
    lv_interface_id = io_edocument->determine_interface_id(
                iv_process_step = lv_process_step ).
  ENDIF.
  IF lv_interface_id IS NOT INITIAL.
    CREATE OBJECT lo_edoc_util.
    TEST-SEAM edoc_intf_conn_trigger_01.
      lv_log_port_name = lo_edoc_util->get_logical_port(
                  iv_interface_id = lv_interface_id
                  iv_bukrs = io_edocument->ms_edocument-bukrs ).
    END-TEST-SEAM.
  ELSE.
*   No Interface is defined for process step &1
    MESSAGE e034(edocument)
      WITH lv_process_step INTO lv_error_txt.
    cl_edocument=>raise_edoc_exception( ).
  ENDIF.
```

```
*   Get AIF interface name and version from the interface ID
   cl_edoc_map_aif=>unpack_interface_id(
    EXPORTING iv_interface_id = lv_interface_id
              iv_int_version  = iv_int_version
    IMPORTING ev_ns         = lv_ns
              ev_ifname     = lv_ifname
              ev_ifversion  = lv_ifversion ).
   ls_interface-ns       = lv_ns.
   ls_interface-ifname   = lv_ifname.
   ls_interface-ifversion = lv_ifversion.

*-- Get target structure
*   Determine the proxy from the request interface
   TEST-SEAM edoc_intf_conn_trigger_02.
    SELECT proxyclassnamecl UP TO 1 ROWS
     FROM /aif/t_finf INTO lv_proxy
     WHERE ns       = ls_interface-ns
      AND ifname    = ls_interface-ifname
      AND ifversion = ls_interface-ifversion.
*    Get the target structure from the response interface
     SELECT SINGLE ddicstructure
      FROM /aif/t_finf INTO lv_target_struc
      WHERE proxyclassnamexi = lv_proxy.
    ENDSELECT.
   END-TEST-SEAM.

   IF sy-subrc <> 0.
    MESSAGE e720(edocument)
     WITH ls_interface-ns ls_interface-ifname
          ls_interface-ifversion
     INTO cl_edocument=>gv_error_txt.
    cl_edocument=>raise_edoc_exception( ).
   ENDIF.
```

```
  TRY.
      CREATE DATA ld_target_struc TYPE (lv_target_struc).
    CATCH cx_sy_create_data_error INTO lx_root.
      cl_edocument=>gv_error_txt = lx_root->get_text( ).
      cl_edocument=>raise_edoc_exception(
        iv_error_txt = cl_edocument=>gv_error_txt ).
  ENDTRY.

  ASSIGN ld_target_struc->* TO <ls_target_structure>.
  IF <ls_source_structure> IS NOT ASSIGNED.
    lo_class_desc = cl_abap_classdescr=>describe_by_object_ref( me ).
    lv_class_name = lo_class_desc->get_relative_name( ).
    MESSAGE e125(edocument)
      WITH 'ld_target_struc->*' '<ls_target_structure>'
           lv_class_name 'TRIGGER'
      INTO cl_edocument=>gv_error_txt.
    cl_edocument=>raise_edoc_exception( ).
  ENDIF.

  CLEAR gt_comm_action.

  IF iv_test_mode IS INITIAL.
*---  Normal mode: AIF interface is created

    CREATE OBJECT lo_edoc_dpp. "2587421
    lo_edoc_dpp->get_bupa(                          "2587421
      EXPORTING iv_edoc_guid = iv_edoc_guid         "2587421
      IMPORTING et_edobupa = lt_edobupa ).          "2587421

*    Trigger AIF interface
    TEST-SEAM edoc_intf_conn_trigger_03.

      CALL FUNCTION 'Z_FI_EDOC_AIF_SEND_CHILE'
        EXPORTING
          ns                = ls_interface-ns
```

```
          ifname             = ls_interface-ifname
          ifversion          = ls_interface-ifversion
          logical_port_name    = lv_log_port_name
        IMPORTING
          ximsgguid          = lv_msgid
        CHANGING
          resp_sap_struct     = <ls_target_structure>
          sap_struct         = <ls_source_structure>
          error_string        = lv_error_txt
        EXCEPTIONS
          persistency_error    = 1
          status_update_failed = 2
          missing_keys        = 3
          OTHERS             = 4.

    END-TEST-SEAM.

    lv_subrc = sy-subrc.

    lv_comm_action = abap_false.

*---  For some AIF errors, FM return success.
*     If ACTION function module was not executed, error ocurred.
*     Check if the action was communicated, if not, then raise an error
    IF NOT lv_msgid IS INITIAL.       "AIF GUID returned
      READ TABLE gt_comm_action
        WITH KEY interface_guid = lv_msgid
        TRANSPORTING NO FIELDS.
      IF sy-subrc <> 0.  "Action has not been communicated for the returned guid
*       Special case: For Interface AIF_XML interface_guid is not maintained
        READ TABLE gt_comm_action
          WITH KEY interface_guid = 'AIF_XML'
                   process_step = lv_process_step
          TRANSPORTING NO FIELDS.
        IF sy-subrc = 0.
```

```
          CLEAR: lv_subrc, lv_error_txt.  "Action was communicated with a dummy guid
        ELSE.
          lv_comm_action = abap_true.
        ENDIF.
      ENDIF.
    ENDIF.

    IF lv_subrc NE 0 OR lv_error_txt IS NOT INITIAL.
      lv_comm_action = abap_true.
    ENDIF.
*---  In case of AIF error ACTION was not called, eDocument need to be updated
*     Update status and AIF GUID if needed
    IF lv_comm_action = abap_true AND NOT lv_msgid IS INITIAL.
*       Determine process step for interface id
      CALL FUNCTION 'EDOC_AIF_GET_INTERFACE_ID'
        IMPORTING
          ev_interface_id = lv_interface_id.

      lv_process_step = io_edocument->determine_process_step(
        EXPORTING iv_interface_id = lv_interface_id ).
      IF lv_process_step IS INITIAL AND iv_interface_id IS INITIAL.
        lv_process_step = 'SENDEDOC'.
      ENDIF.
*       Update eDocument with information about AIF interface executed
*       Call the communicate process step only if it has not been called
*       yet for the current step
*       Get eDocument instance with the last eDocument data
      cl_edocument=>retrieve_by_edoc_guid(
        EXPORTING iv_edoc_guid = iv_edoc_guid
        RECEIVING ro_edocument = lo_edocument ).

      lv_if_guid = lv_msgid.
      IF lv_process_step = 'NONE'.
        lo_edocument->update_interface_guid(
          iv_interface_guid = lv_if_guid ).
```

```
      ELSE.
       GET REFERENCE OF <ls_target_structure> INTO ld_aif_data.
       lo_intf_conn_impl->get_process_step_data(
        EXPORTING
         iv_process_step = lv_process_step
         ir_data = ld_aif_data
         iv_process = lo_edocument->ms_edocument-process
         iv_process_ver = lo_edocument->ms_edocument-process_version
        IMPORTING
         er_data = ld_process_data ).

       lo_edocument->communicate_process_step(
        iv_process_step   = lv_process_step
        iv_interface_guid = lv_if_guid
        id_data           = ld_process_data
        iv_error          = abap_true                "2700218
         ).
      ENDIF.
    ENDIF.

*   AIF interface failed, then raise an error
    IF ( lv_subrc NE 0 OR lv_error_txt IS NOT INITIAL )
      OR lv_comm_action = abap_true.

      lo_edoc_dpp->reset_bupa(                       "2587421
       EXPORTING iv_edoc_guid = iv_edoc_guid          "2587421
             it_edobupa   = lt_edobupa ).          "2587421

*     Error triggering message, check AIF error log for details
      MESSAGE e018(edocument)
        INTO lv_error_txt.

      cl_edocument=>raise_edoc_exception( ).

    ENDIF.
```

```
  ELSE.
*---  Test mode: AIF mapping is executed to find errors
    CALL FUNCTION '/AIF/FILE_TRANSFORM_DATA'
      EXPORTING
       ns                  = ls_interface-ns
       ifname              = ls_interface-ifname
       ifversion           = ls_interface-ifversion
       ifdirection         = 'O'   "Outbound
*    IMPORTING
*       out_struct          = ls_proxy_input
      CHANGING
       raw_struct          = <ls_source_structure>
      EXCEPTIONS
       not_found           = 1
       customizing_incomplete = 2
       max_errors_reached    = 3
       cancel              = 4
       OTHERS              = 5.
    IF sy-subrc <> 0.
*    Propagate error contained in SY structure
      IF NOT sy-msgno IS INITIAL AND
        NOT sy-msgid IS INITIAL.

       cl_edocument=>raise_edoc_exception( ).

      ELSE.
*      Propagate generic error
*      Error triggering message, check AIF error log for details
       MESSAGE e018(edocument)
         INTO lv_error_txt.

       cl_edocument=>raise_edoc_exception( ).

      ENDIF.
```

```
    ENDIF.
  ENDIF.
*   ENDIF.
*  ENDMETHOD.
 ENDMETHOD.
```

Appendix - 2

Function Module 1

```
FUNCTION FUNC_1.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(NS) TYPE  /AIF/NS OPTIONAL
*"     REFERENCE(IFNAME) TYPE  /AIF/IFNAME OPTIONAL
*"     REFERENCE(IFVERSION) TYPE  /AIF/IFVERSION OPTIONAL
```

```
*"    REFERENCE(APPL_LOG_FUNCTION) TYPE  /AIF/IFACTION
*"       DEFAULT 'AIF_SEND_WITH_PROXY'
*"    REFERENCE(DO_COMMIT) TYPE  CHAR01 DEFAULT 'X'
*"    REFERENCE(RESTART_MSG_GUID) TYPE  SXMSMGUID OPTIONAL
*"    REFERENCE(MSGGUID) TYPE  SXMSMGUID OPTIONAL
*"    REFERENCE(IS_RESTART) TYPE  CHAR01 OPTIONAL
*"    REFERENCE(IV_PERSIST_XML) TYPE  CHAR01 DEFAULT 'X'
*"    REFERENCE(LOGICAL_PORT_NAME) TYPE  PRX_LOGICAL_PORT_NAME
*"       OPTIONAL
*"    REFERENCE(IV_EOIO) TYPE  ABAP_BOOL OPTIONAL
*"    REFERENCE(IV_QUEUE_ID) TYPE  PRX_SCNT OPTIONAL
*"  EXPORTING
*"    REFERENCE(XIMSGGUID) TYPE  SXMSMGUID
*"    REFERENCE(XIMSGGUID_OUT) TYPE  SXMSMGUID
*"  TABLES
*"     ADD_RETURN_TAB STRUCTURE  BAPIRET2 OPTIONAL
*"  CHANGING
*"    REFERENCE(RESP_SAP_STRUCT) OPTIONAL
*"    REFERENCE(SAP_STRUCT) OPTIONAL
*"    REFERENCE(ERROR_STRING) TYPE  STRING OPTIONAL
*"    REFERENCE(ERROR_LONG_TEXT) TYPE  STRING OPTIONAL
*"    REFERENCE(REF_TO_PROXY) TYPE REF TO OBJECT OPTIONAL
*"  EXCEPTIONS
*"     PERSISTENCY_ERROR
*"     STATUS_UPDATE_FAILED
*"     MISSING_KEYS
*"     INTERFACE_NOT_FOUND
*"     TRANSFORMATION_ERROR
*"     GENERAL_ERROR
*"----------------------------------------------------------------------


 DATA: lr_appl_engine TYPE REF TO /aif/if_application_engine,
       lr_xmlparse    TYPE REF TO data,
       ls_xmlparse    TYPE /aif/xmlparse_data, " for storing data
       ls_xmlparse_r  TYPE /aif/xmlparse_data, " for retrieving data (indicated by "_r")
```

```
    lr_sap_struct  TYPE REF TO data,
    ls_sy_old      TYPE syst,
    lr_resp_raw_struct  TYPE REF TO data,
    ls_outbound_finf    TYPE /aif/t_finf,
    ls_inbound_finf     TYPE /aif/t_finf.

DATA:
 lr_t100        TYPE REF TO if_t100_message,
 lr_static_error TYPE REF TO cx_static_check,
 lv_msg_locked   TYPE abap_bool,
 lv_mandt        TYPE syst-mandt,
 lv_msgguid      TYPE guid_32.

FIELD-SYMBOLS: <ls_sap_struct> TYPE ANY.

IF ref_to_proxy IS NOT INITIAL AND logical_port_name IS NOT INITIAL.
 MESSAGE ID '/AIF/MES' TYPE 'E' NUMBER 530
         RAISING general_error.
ENDIF.

TRY.
 CALL METHOD /aif/cl_aif_engine_factory=>get_engine
 EXPORTING
  iv_ns          = ns
  iv_ifname      = ifname
  iv_ifversion   = ifversion
 RECEIVING
  rref_appl_engine = lr_appl_engine.
CATCH /aif/cx_aif_engine_not_found.
 RAISE persistency_error.
CATCH /aif/cx_error_handling_general.
 RAISE persistency_error.
ENDTRY.

CLEAR ximsgguid_out.
```

```
IF NOT msgguid IS INITIAL AND NOT sap_struct IS SUPPLIED.
  " try to load message data (sap structure) from XML persistency

  TRY.
    CALL METHOD lr_appl_engine->read_msg_from_persistency
      EXPORTING
        iv_msgguid  = msgguid
        iv_ns       = ns
        iv_ifname   = ifname
        iv_ifver    = ifversion
      CHANGING
      cref_data                 = lr_sap_struct
      cs_xmlparse               = ls_xmlparse_r
      .
    CATCH /aif/cx_error_handling_general .
    RAISE persistency_error.
  ENDTRY.

  ASSIGN lr_sap_struct->* TO <ls_sap_struct>.
  IF NOT sy-subrc IS INITIAL.
    RAISE persistency_error. " TODO: raise exception -> data could not be loaded
  ENDIF.
*   sap_struct is not supplied
*    sap_struct = <ls_sap_struct>.
 ELSE.
  ASSIGN sap_struct TO <ls_sap_struct>.
 ENDIF.

 CALL FUNCTION 'FUNC_2'
  EXPORTING
    ns                = ns
    ifname            = ifname
    ifversion         = ifversion
    appl_log_function    = appl_log_function
    do_commit         = do_commit
```

```
    restart_msg_guid     = restart_msg_guid
    msgguid              = msgguid
    is_restart           = is_restart
    logical_port_name    = logical_port_name
    iv_eoio              = iv_eoio
    iv_queue_id          = iv_queue_id
  IMPORTING
    ximsgguid            = ximsgguid
    ximsgguid_out        = ximsgguid_out
    es_inbound_finf      = ls_inbound_finf
    es_outbound_finf     = ls_outbound_finf
  TABLES
    add_return_tab       = add_return_tab
  CHANGING
    resp_sap_struct      = resp_sap_struct
    sap_struct           = <ls_sap_struct>
    error_string         = error_string
    error_long_text      = error_long_text
    resp_raw_struct      = lr_resp_raw_struct
    ref_to_proxy         = ref_to_proxy
  EXCEPTIONS
    persistency_error    = 1
    status_update_failed = 2
    interface_not_found  = 3
    transformation_error = 4
    general_error        = 5
    OTHERS               = 6.

 ls_sy_old = sy.

 IF iv_persist_xml = 'X'.
   TRY.
*   Delete older persistency data
   IF NOT msgguid IS INITIAL .
        lr_appl_engine->delete_msg_from_persistency(
```

```
                    EXPORTING iv_msgguid    = msgguid
                        iv_pipelineid = 'SENDER' '' NOT USED
                        iv_ns        =  ns
                        iv_ifname     = ifname
                        iv_ifver      = ifversion
                      ).
  ENDIF.
  IF ximsgguid <> ximsgguid_out AND ximsgguid_out IS NOT INITIAL.
    "outbound: persist the request data
    ls_xmlparse-msgguid = ximsgguid_out.
*     Note 2144871: Set the outbound interface, use the interface returned from /AIF/SEND_WITH_PROXY_CORE
*     as it might has been determined via interface determination
    ls_xmlparse-ns = ls_outbound_finf-ns.
    ls_xmlparse-ifname = ls_outbound_finf-ifname.
    ls_xmlparse-ifver = ls_outbound_finf-ifversion.
    GET REFERENCE OF <ls_sap_struct>  INTO ls_xmlparse-xi_data.
    GET REFERENCE OF ls_xmlparse      INTO lr_xmlparse.
    lr_appl_engine->persist_message_data( CHANGING cr_xmlparse = lr_xmlparse ).
    "inbound: persist the response data
*     Note 2144871: Set the outbound interface, use the interface returned from /AIF/SEND_WITH_PROXY_CORE
*     as it might has been determined via interface determination
    CLEAR ls_xmlparse.
    ls_xmlparse-msgguid = ximsgguid.
    ls_xmlparse-ns = ls_inbound_finf-ns.
    ls_xmlparse-ifname = ls_inbound_finf-ifname.
    ls_xmlparse-ifver = ls_inbound_finf-ifversion.
    ls_xmlparse-xi_data = lr_resp_raw_struct.
    GET REFERENCE OF ls_xmlparse INTO lr_xmlparse.

    lr_appl_engine->persist_message_data( CHANGING cr_xmlparse = lr_xmlparse ).
  ELSEIF ximsgguid = ximsgguid_out OR ximsgguid_out IS INITIAL.
    ls_xmlparse-msgguid = ximsgguid.
    ls_xmlparse-ns      = ls_outbound_finf-ns.
    ls_xmlparse-ifname  = ls_outbound_finf-ifname.
    ls_xmlparse-ifver   = ls_outbound_finf-ifversion.
```

```
 GET REFERENCE OF <ls_sap_struct>  INTO ls_xmlparse-xi_data.
 GET REFERENCE OF ls_xmlparse      INTO lr_xmlparse.
 lr_appl_engine->persist_message_data( CHANGING cr_xmlparse = lr_xmlparse ).


ENDIF.
 CATCH /aif/cx_engine_xml /aif/cx_error_handling_general INTO lr_static_error.
   lv_mandt = lv_mandt = cl_abap_syst=>get_client( ).
   CLEAR lv_msgguid.
   IF msgguid IS NOT INITIAL.
    lv_msgguid = msgguid.
   ELSEIF ximsgguid_out IS INITIAL.
    lv_msgguid = ximsgguid.
   ENDIF.
   IF lv_msgguid IS NOT INITIAL.
     CALL FUNCTION 'DEQUEUE_/AIF/EMSGGUID'
       EXPORTING
        mode_/aif/msgguid_st = 'E'
        mandt              = lv_mandt
        msgguid            = lv_msgguid
        _scope             = 1
       EXCEPTIONS
        error_message      = 1
        OTHERS             = 2.
     IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE 'S' NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
     ENDIF.
   ENDIF.
   IF ximsgguid_out IS NOT INITIAL.
     lv_msgguid = ximsgguid_out.
     CALL FUNCTION 'DEQUEUE_/AIF/EMSGGUID'
       EXPORTING
        mode_/aif/msgguid_st = 'E'
        mandt              = lv_mandt
        msgguid            = lv_msgguid
```

```
          _scope              = 1
        EXCEPTIONS
          error_message       = 1
          OTHERS              = 2.
      IF sy-subrc <> 0.
        MESSAGE ID sy-msgid TYPE 'S' NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
      ENDIF.
    ENDIF.
    "     /aif/cx_engine_xml and /aif/cx_error_Hanlding_general have interface if_t100_message
    lr_t100 = CAST #( lr_static_error ).
    MESSAGE ID lr_t100->t100key-msgid TYPE 'E' NUMBER sy-msgno
      WITH lr_t100->t100key-attr1
           lr_t100->t100key-attr2
           lr_t100->t100key-attr3
           lr_t100->t100key-attr4
      RAISING persistency_error.
  ENDTRY.
ENDIF.
IF do_commit = abap_true.
  COMMIT WORK AND WAIT.
ENDIF.
lv_mandt = lv_mandt = cl_abap_syst=>get_client( ).
CLEAR lv_msgguid.
IF msgguid IS NOT INITIAL.
  lv_msgguid = msgguid.
ELSEIF ximsgguid_out IS INITIAL.
  lv_msgguid = ximsgguid.
ENDIF.
IF lv_msgguid IS NOT INITIAL.
 CALL FUNCTION 'DEQUEUE_/AIF/EMSGGUID'
   EXPORTING
     mode_/aif/msgguid_st = 'E'
     mandt              = lv_mandt
     msgguid            = lv_msgguid
```

```
      _scope              = 1
    EXCEPTIONS
      error_message       = 1
      OTHERS              = 2.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE 'S' NUMBER sy-msgno
          WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
ENDIF.
IF ximsgguid_out IS NOT INITIAL.
  lv_msgguid = ximsgguid_out.
  CALL FUNCTION 'DEQUEUE_/AIF/EMSGGUID'
    EXPORTING
      mode_/aif/msgguid_st = 'E'
      mandt              = lv_mandt
      msgguid            = lv_msgguid
      _scope             = 1
    EXCEPTIONS
      error_message       = 1
      OTHERS              = 2.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE 'S' NUMBER sy-msgno
          WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
ENDIF.



CASE ls_sy_old-subrc.
  WHEN 1.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 RAISING persistency_error.
  WHEN 2.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 RAISING status_update_failed.
  WHEN 3.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 RAISING interface_not_found.
  WHEN 4.
```

```
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 RAISING transformation_error.
  WHEN 5.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 RAISING general_error.
 ENDCASE.

ENDFUNCTION.
```

Function Module 2

```
FUNCTION FUNC_2.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(NS) TYPE  /AIF/NS OPTIONAL
*"     REFERENCE(IFNAME) TYPE  /AIF/IFNAME OPTIONAL
*"     REFERENCE(IFVERSION) TYPE  /AIF/IFVERSION OPTIONAL
*"     REFERENCE(APPL_LOG_FUNCTION) TYPE  /AIF/IFACTION DEFAULT
*"       'AIF_SEND_WITH_PROXY'
*"     REFERENCE(DO_COMMIT) TYPE  CHAR01 DEFAULT 'X'
*"     REFERENCE(RESTART_MSG_GUID) TYPE  SXMSMGUID OPTIONAL
*"     REFERENCE(MSGGUID) TYPE  SXMSMGUID OPTIONAL
*"     REFERENCE(IS_RESTART) TYPE  CHAR01 OPTIONAL
*"     REFERENCE(LOGICAL_PORT_NAME) TYPE  PRX_LOGICAL_PORT_NAME
*"       OPTIONAL
*"     REFERENCE(IV_EOIO) TYPE  ABAP_BOOL OPTIONAL
*"     REFERENCE(IV_QUEUE_ID) TYPE  PRX_SCNT OPTIONAL
*"  EXPORTING
*"     REFERENCE(XIMSGGUID) TYPE  SXMSMGUID
*"     REFERENCE(XIMSGGUID_OUT) TYPE  SXMSMGUID
*"     REFERENCE(ES_INBOUND_FINF) TYPE  /AIF/T_FINF
*"     REFERENCE(ES_OUTBOUND_FINF) TYPE  /AIF/T_FINF
*"  TABLES
*"      ADD_RETURN_TAB STRUCTURE  BAPIRET2 OPTIONAL
*"  CHANGING
*"     REFERENCE(RESP_SAP_STRUCT) OPTIONAL
*"     REFERENCE(SAP_STRUCT) OPTIONAL
*"     REFERENCE(ERROR_STRING) TYPE  STRING OPTIONAL
```

```
*"    REFERENCE(ERROR_LONG_TEXT) TYPE  STRING OPTIONAL
*"    REFERENCE(RESP_RAW_STRUCT) TYPE REF TO  DATA OPTIONAL
*"    REFERENCE(REF_TO_PROXY) TYPE REF TO  OBJECT OPTIONAL
*"  EXCEPTIONS
*"     PERSISTENCY_ERROR
*"     STATUS_UPDATE_FAILED
*"     INTERFACE_NOT_FOUND
*"     TRANSFORMATION_ERROR
*"     GENERAL_ERROR
*"----------------------------------------------------------------------


 DATA: lr_enabler               TYPE REF TO /aif/cl_enabler_proxy,
       ls_xi_data               TYPE REF TO data,
       ls_xi_data_input         TYPE REF TO data,

       ls_finf                  TYPE /aif/t_finf,
       ls_ifkeys                TYPE /aif/ifkeys,
       ls_variant               TYPE /aif/t_variant,

       lr_proxy                 TYPE REF TO object,
       lr_prot                  TYPE REF TO if_wsprotocol,
       lr_msgid_prot            TYPE REF TO if_wsprotocol_message_id,
       lx_exc                   TYPE REF TO cx_root,

       lv_methodname            TYPE abap_methname,
       lv_ddicstruct_input      TYPE tabname,
       lv_rectype_input         TYPE /aif/rectyperaw,
       lv_ddicstruct_output     TYPE tabname,
       lv_rectype_output        TYPE /aif/rectyperaw,

       lv_current_msg_state     TYPE /aif/proc_status,
       lv_current_message_location TYPE /aif/message_location,
       lv_ximsgguid_cast        TYPE guid_32,

       ls_inbound_finf          TYPE /aif/t_finf,
```

```
lv_new_guid             TYPE guid_32,

lt_ifacts               TYPE STANDARD TABLE OF /aif/t_ifact,
lt_actions              TYPE STANDARD TABLE OF /aif/t_act,
lt_actionts             TYPE STANDARD TABLE OF /aif/t_actt,
lt_funcs                TYPE STANDARD TABLE OF /aif/t_func,
lt_accheck              TYPE STANDARD TABLE OF /aif/t_accheck,
lt_check                TYPE STANDARD TABLE OF /aif/t_check,
lt_tabchk               TYPE STANDARD TABLE OF /aif/t_tabchk,
lt_tabfld               TYPE STANDARD TABLE OF /aif/t_tabfld,
lt_acfields             TYPE STANDARD TABLE OF /aif/t_acfields,
lt_intrec               TYPE STANDARD TABLE OF /aif/t_intrec,
lt_recflds              TYPE STANDARD TABLE OF /aif/t_recflds,
lt_fault                TYPE STANDARD TABLE OF bapiret2,
lt_temp_return          TYPE STANDARD TABLE OF bapiret2,
lv_successflag          TYPE /aif/successflag,
lt_hash_tab             TYPE /aif/hashtab_t,
lt_idx_tab              TYPE /aif/idxtab_t,
ls_sproxhdr             TYPE sproxhdr,
lv_sync                 TYPE c VALUE '',

lr_msgid_internal       TYPE REF TO if_wsprotocol_internal,
lr_xmb                  TYPE REF TO cl_xms_message_xmb,
lr_runtime              TYPE REF TO cl_xms_run_time_env,
lv_pipelineid           TYPE sxmspid,
lv_pipelineid_ext       TYPE sxmspidext,

lt_message_return       TYPE bapiret2_t,  " this will be cleared after messages are added to enabler buffer
lt_all_bal_messages     TYPE /aif/bal_t_msg,
ls_bal_context          TYPE /aif/bal_context,
lt_log_handle_save      TYPE bal_t_logh,
lv_message_no           TYPE symsgno,
lv_dummy_message        TYPE string,     " for where used list

lv_lines                TYPE int4,
```

```
      lv_2nd_langu              TYPE sylangu,
      lv_fb_langu               TYPE sylangu,
      ls_sy_old                 TYPE sy,
      lv_subrc                  TYPE sysubrc,

      lv_error                  TYPE c,
      lv_error_all              TYPE c.


  DATA: lt_out_ddic_tree TYPE /aif/ddic_tree_t,
      gv_msgdate      TYPE   syst_datum,
      gv_msgtime      TYPE   syst_uzeit,
      gv_ns           TYPE  /aif/ns,
      gv_ifname       TYPE  /aif/ifname,
      gv_ifversion    TYPE  /aif/ifversion,
      gs_finf         TYPE  /aif/t_finf,
      ls_str          TYPE edoc_cl_mapping_source. "Note 1877297
  DATA lv_tabname TYPE tabname. "Note 1877297
  DATA: lr_protocol       TYPE REF TO if_wsprotocol,
      lr_async_messaging TYPE REF TO if_wsprotocol_async_messaging.
  FIELD-SYMBOLS: <lrs_xi_param>, <lrs_xi_data>, <lrs_xi_param_input>, <lrs_xi_data_input>.
  DATA lr_runtime_badi TYPE REF TO /aif/runtime_methods. "Note 2151693
  DATA lv_guid_32 TYPE guid_32. "Note 2151693
  DATA lv_guid_32_temp TYPE guid_32. "Note 2151693
  DATA lv_datum       TYPE sydatum.
  DATA lv_uzeit       TYPE syuzeit.
* generic call of proxy method
  DATA: lt_parmbind     TYPE abap_parmbind_tab,
      ls_parmbind_line TYPE abap_parmbind,
      lt_sproxdat      TYPE TABLE OF sproxdat,
      ls_sproxdat      TYPE sproxdat.
  FIELD-SYMBOLS: <fs_sproxdat> TYPE sproxdat.

  DATA:
    lv_msg_locked TYPE abap_bool,
```

```
  lv_mandt      TYPE syst-mandt.


*  DBA manager
 DATA: lr_dba_mgr TYPE REF TO /aif/cl_dba_reader_mgr.
 TRY.
    lr_enabler = /aif/cl_enabler_proxy=>build(
       is_raw_structure   = sap_struct
       iv_ns            = ns
       iv_ifname         = ifname
       iv_ifversion      = ifversion
       iv_msgguid        = msgguid
       is_ws_add_proxy    = VALUE /aif/add_proxy_s( loc_indicator = /aif/cl_passport_service=>mc_location_indicator-client
                                    logical_port = logical_port_name  )
                                    ).


  CATCH /aif/cx_enabler_proxy .
    IF lr_enabler IS INITIAL.
      MESSAGE e040(/aif/error_handling) WITH ns ifname ifversion RAISING interface_not_found.
    ENDIF.
 ENDTRY.


 lv_ximsgguid_cast = lr_enabler->get_msgguid( ).
 ximsgguid        = lv_ximsgguid_cast.
 lv_guid_32_temp   = lv_ximsgguid_cast."Note 2151693


*  gv_msgguid       = ximsgguid."Note 2151669


 gv_msgdate    =  sy-datum."Note 2151669
 gv_msgtime    =  sy-uzeit."Note 2151669


 lv_mandt = cl_abap_syst=>get_client( ).
 DO 5 TIMES.
  CALL FUNCTION 'ENQUEUE_/AIF/EMSGGUID'
    EXPORTING
      mode_/aif/msgguid_st = 'E'
```

```
      mandt              = lv_mandt
      _scope             = '1' "in program scope
      msgguid            = lv_ximsgguid_cast
    EXCEPTIONS
      foreign_lock       = 1
      system_failure     = 2
      OTHERS             = 3.
  IF sy-subrc <> 0.
    lv_msg_locked = abap_false.
    WAIT UP TO 1 SECONDS.
  ELSE.
    lv_msg_locked = abap_true.
    EXIT.
  ENDIF.
 ENDDO.
 IF lv_msg_locked = abap_false.
  MESSAGE e028(/aif/runtime) WITH lv_guid_32_temp RAISING general_error.
 ENDIF.
**********Note 2007548 Set outbound PID*************
 lr_enabler->set_pid('AIF_SENDER').
**********Note 2007548 Set outbound PID*************

* 941ff. save additional messages given by calling function
 CLEAR ls_bal_context.
 ls_bal_context-msg_category = 'O'.
 ls_bal_context-ifaction     = appl_log_function.

 lt_message_return[] = add_return_tab[].
 lr_enabler->add_bapiret2_messages(
  it_bapiret2_messages = lt_message_return
  is_bal_context = ls_bal_context
  iv_msg_source = /aif/if_globals=>gc_eh_msg_source-interface ).

 " 'E' is the most likely status for error situations - so read this one first
 READ TABLE lt_message_return WITH KEY type = 'E' TRANSPORTING NO FIELDS.
```

```
IF NOT sy-subrc IS INITIAL.
  READ TABLE lt_message_return WITH KEY type = 'A' TRANSPORTING NO FIELDS.
ENDIF.
IF sy-subrc IS INITIAL.
  TRY.
      " Do not explicitly set message status here - final status will be derived from buffer filled before with add_bepiret2_messages
      lr_enabler->/aif/if_enabler_base~update(
        iv_message_aif_location = /aif/if_globals=>gc_message_location-before_aif
        is_sap_structure       = sap_struct
        iv_logical_port        = logical_port_name
        iv_do_commit           = do_commit ).
    CATCH /aif/cx_enabler_base .
      MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
  ENDTRY.
  RETURN. " final error status was set - bye bye
ELSE.
  TRY.
      " messages have been added to instance-internal buffer before and need not be provided here
      lr_enabler->/aif/if_enabler_base~update(
        iv_message_status_flag  = 'I'
        iv_message_aif_location = /aif/if_globals=>gc_message_location-before_aif
        is_sap_structure       = sap_struct
        iv_logical_port        = logical_port_name
        iv_do_commit           = do_commit ).
    CATCH /aif/cx_enabler_base .
      MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
  ENDTRY.
  CLEAR lt_message_return.
ENDIF.


ls_finf = lr_enabler->get_finf( ).
ls_ifkeys-ns     = ls_finf-ns.
ls_ifkeys-ifname = ls_finf-ifname.
ls_ifkeys-ifver  = ls_finf-ifversion.
es_outbound_finf = ls_finf. "Note 2144871: return the outbound interface, so that it can be persisted with the data
```

```
gv_ns        = ls_finf-ns.
gv_ifname    = ls_finf-ifname.
gv_ifversion  = ls_finf-ifversion.
gs_finf      = ls_finf.

/aif/cl_proxy_outbound=>set_current_tmp_msgguid( iv_tmp_msgguid = ximsgguid ).

/aif/cl_proxy_outbound=>deactivate_implicit_enabler(
                        iv_ns       = gv_ns
                        iv_ifname    = gv_ifname
                        iv_version   = gv_ifversion
                        iv_ximsgguid = ximsgguid ).

CALL FUNCTION '/AIF/DETERMINE_VARIANT'
 EXPORTING
   ir_rawdata = sap_struct
   is_finf    = ls_finf
 IMPORTING
   es_variant = ls_variant.

CALL FUNCTION '/AIF/AUTH_CHECK_PROCESSING'
 EXPORTING
   iv_ns     = ls_finf-ns
   iv_ifname = ls_finf-ifname
   iv_ifver  = ls_finf-ifversion
   iv_vns    = ls_variant-variant_ns
   iv_vname  = ls_variant-variant_name
   iv_action = 'O'
 IMPORTING
   rc       = lv_subrc.
IF NOT lv_subrc IS INITIAL.
 MESSAGE e160(/aif/mes) WITH TEXT-a03  INTO lv_dummy_message. " where used list

 lr_enabler->add_single_log_message( iv_msg_type       = 'E'
```

```
                        iv_msg_id        = '/AIF/MES'
                        iv_msg_number    = '160'
                        iv_msg_message_v1 = TEXT-a03
                        iv_msg_message   = lv_dummy_message
                        is_bal_context   = ls_bal_context ).
  TRY.
     lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'E' iv_do_commit = do_commit ).
    CATCH /aif/cx_enabler_base .
     MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
  ENDTRY.
  RETURN.
 ENDIF.

 lr_dba_mgr = /aif/cl_dba_reader_mgr=>get_instance( ).

* 470ff. get all actions, functions and fix values
 DATA:lr_dba_ifact TYPE REF TO /aif/if_dba_ifact,
     ls_ifact_key TYPE /aif/t_ifact.

 lr_dba_ifact = lr_dba_mgr->get_dba_ifact( ).
 ls_ifact_key-ns = ns.
 ls_ifact_key-ifname = ifname.
 ls_ifact_key-ifversion = ifversion.
 IF lr_dba_ifact IS BOUND.
  TRY.
     lt_ifacts = lr_dba_ifact->read_table( is_key = ls_ifact_key ).
     SORT lt_ifacts BY actionnr.
    CATCH /aif/cx_dba_reader.  "
     CLEAR lt_ifacts.
  ENDTRY.
 ENDIF.

 IF  lt_ifacts IS NOT INITIAL.

  DATA:
```

```
  lr_dba_act TYPE REF TO /aif/if_dba_act,
  lt_act_key TYPE /aif/tt_act_keys,
  ls_act_key TYPE /aif/s_act_keys.

FIELD-SYMBOLS: <ls_ifact> LIKE LINE OF lt_ifacts.



lr_dba_act ?= lr_dba_mgr->get_dba_act( ).

IF lr_dba_act IS BOUND.

  LOOP AT lt_ifacts ASSIGNING <ls_ifact>.
    ls_act_key-ns = <ls_ifact>-nsaction.
    ls_act_key-ifaction = <ls_ifact>-ifaction.
    INSERT ls_act_key INTO TABLE lt_act_key.
  ENDLOOP.

  TRY.
      lt_actions = lr_dba_act->read_table_fae_keys_deli_all( it_keys = lt_act_key ).
    CATCH /aif/cx_dba_reader.
      CLEAR lt_actions.
  ENDTRY.

  " language support
  lv_2nd_langu = /aif/cl_lang_control=>return_2nd_lang( ).
  lv_fb_langu  = /aif/cl_lang_control=>return_fb_lang( ).

  lt_actionts = lr_dba_act->read_texts_fae_keys_by_langu( it_keys  = lt_act_key ).

  IF lt_actionts[] IS INITIAL AND lv_2nd_langu IS NOT INITIAL.
    lt_actionts = lr_dba_act->read_texts_fae_keys_by_langu( it_keys  = lt_act_key iv_langu = lv_2nd_langu ).
  ENDIF.

  IF lt_actionts[] IS INITIAL AND lv_fb_langu IS NOT INITIAL.
```

```
      lt_actionts = lr_dba_act->read_texts_fae_keys_by_langu( it_keys  = lt_act_key iv_langu = lv_fb_langu ).
    ENDIF.

    IF lt_actionts[] IS INITIAL.
      lt_actionts = lr_dba_act->read_texts_fae_keys_by_langu( it_keys  = lt_act_key iv_ignore_langu = abap_true ).
    ENDIF.

  ENDIF.
 ENDIF.

 IF NOT lt_actions[] IS INITIAL.
  SELECT * FROM /aif/t_func INTO TABLE lt_funcs FOR ALL ENTRIES IN lt_actions
    WHERE ns = lt_actions-ns AND ifaction = lt_actions-ifaction.
  SORT lt_funcs ASCENDING BY funcnr.
 ENDIF.
 IF lt_funcs[] IS NOT INITIAL.
  SELECT * FROM /aif/t_accheck INTO TABLE lt_accheck FOR ALL ENTRIES IN lt_funcs
    WHERE ns = lt_funcs-ns AND ifaction = lt_funcs-ifaction AND funcnr = lt_funcs-funcnr.
  SORT lt_accheck BY ns ifaction funcnr checknr.
  IF lt_accheck[] IS NOT INITIAL.
    SELECT * INTO TABLE lt_check FROM /aif/t_check FOR ALL ENTRIES IN lt_accheck
      WHERE ns = lt_accheck-nscheck AND aifcheck = lt_accheck-aifcheck.

    SELECT * INTO TABLE lt_tabchk FROM /aif/t_tabchk FOR ALL ENTRIES IN lt_accheck
      WHERE ns = lt_accheck-nscheck AND aifcheck = lt_accheck-aifcheck.
    SORT lt_tabchk BY ns aifcheck checknr.
  ENDIF.
 ENDIF.

* 584ff. get name of method and data element used for proxy
** Get the Synchron field value from the Table SPROXDAT for the Given Proxy class.
 IF ls_finf-prx_method IS NOT INITIAL.
* Proxy Method is Provided
   SELECT SINGLE synchron FROM sproxdat INTO lv_sync
    WHERE object = 'CLAS' AND obj_name = ls_finf-proxyclassnamecl
```

```
    AND object1 = 'METH' AND obj_name1 = ls_finf-prx_method AND object2 = 'METH' AND inactive = ''.


   IF sy-subrc NE 0.
    MESSAGE a521(/aif/mes) WITH ls_finf-proxyclassnamecl ls_finf-prx_method INTO lv_dummy_message. " where used list


    lr_enabler->add_single_log_message( iv_msg_type      = 'A'
                                iv_msg_id        = '/AIF/MES'
                                iv_msg_number    = '521'
                                iv_msg_message_v1 = ls_finf-proxyclassnamecl
                                iv_msg_message_v2 = ls_finf-prx_method
                                iv_msg_message    = lv_dummy_message
                                is_bal_context    = ls_bal_context ).


    TRY.
       lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'A' iv_do_commit = do_commit ).
      CATCH /aif/cx_enabler_base .
       MESSAGE a197(/aif/error_handling) RAISING status_update_failed.
    ENDTRY.
    RETURN.
   ENDIF.


  ELSE.


* Proxy Method is Not Provided . In this case, get the fisrt entry from the table
* For the given Proxy Class
   SELECT SINGLE synchron FROM sproxdat INTO lv_sync
    WHERE object = 'CLAS' AND obj_name = ls_finf-proxyclassnamecl
    AND object1 = 'METH' AND object2 = 'METH' AND inactive = ''.


   IF sy-subrc NE 0.
    MESSAGE a522(/aif/mes) WITH ls_finf-proxyclassnamecl INTO lv_dummy_message. " where used list


    lr_enabler->add_single_log_message( iv_msg_type      = 'A'
                                iv_msg_id        = '/AIF/MES'
                                iv_msg_number    = '522'
```

```
                        iv_msg_message_v1 = ls_finf-proxyclassnamecl
                        iv_msg_message    = lv_dummy_message
                        is_bal_context    = ls_bal_context ).
    TRY.
      lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'A' iv_do_commit = do_commit ).
     CATCH /aif/cx_enabler_base .
      MESSAGE a197(/aif/error_handling) RAISING status_update_failed.
    ENDTRY.
    RETURN.

  ENDIF.

 ENDIF.

* 684ff get method names
 TRY.
    /aif/cl_enabler_proxy=>get_proxyclass_infos(
     EXPORTING
      iv_proxy_classname   = ls_finf-proxyclassnamecl
      iv_proxy_method      = ls_finf-prx_method
     IMPORTING
      ev_methodname        = lv_methodname
      ev_ddicstruct_input  = lv_ddicstruct_output
      ev_rectype_input     = lv_rectype_output
      ev_ddicstruct_output = lv_ddicstruct_input
      ev_rectype_output    = lv_rectype_input ).
    lv_rectype_output = ls_finf-rectyperaw.
   CATCH /aif/cx_enabler_proxy .
    "TODO: handle errors
*    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 RAISING cancel.
 ENDTRY.


* 738: create variable for XI structure (for OUTPUT)
 CREATE DATA ls_xi_data TYPE (lv_ddicstruct_output).
```

```
IF NOT sy-subrc IS INITIAL.
  MESSAGE e085(/aif/mes) WITH lv_ddicstruct_output INTO lv_dummy_message. " where used list

  lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                      iv_msg_id        = '/AIF/MES'
                                      iv_msg_number    = '085'
                                      iv_msg_message_v1 = lv_ddicstruct_output
                                      iv_msg_message   = lv_dummy_message
                                      is_bal_context   = ls_bal_context ).
  TRY.
      lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'E' iv_do_commit = do_commit ).
    CATCH /aif/cx_enabler_base .
      MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
  ENDTRY.
  RETURN.
ENDIF.

IF lv_rectype_output IS INITIAL.
  ASSIGN ls_xi_data->* TO <lrs_xi_data>.
ELSE.
  ASSIGN ls_xi_data->(lv_rectype_output) TO <lrs_xi_data>.
ENDIF.
IF sy-subrc IS INITIAL.
  ASSIGN ls_xi_data->* TO <lrs_xi_param>.
ENDIF.
IF NOT sy-subrc IS INITIAL.
  MESSAGE e074(/aif/mes) INTO lv_dummy_message. " where used list
  lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                      iv_msg_id        = '/AIF/MES'
                                      iv_msg_number    = '074'
                                      iv_msg_message   = lv_dummy_message
                                      is_bal_context   = ls_bal_context ).
  TRY.
      lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'E' iv_do_commit = do_commit ).
    CATCH /aif/cx_enabler_base .
```

```
      MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
    ENDTRY.
    RETURN.
  ENDIF.

* 821ff create variable for XI structure (for INPUT)
  IF NOT lv_sync IS INITIAL.
    CREATE DATA ls_xi_data_input TYPE (lv_ddicstruct_input).
    IF NOT sy-subrc IS INITIAL.
      MESSAGE e085(/aif/mes) WITH lv_ddicstruct_input INTO lv_dummy_message. " where used list

      lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                iv_msg_id        = '/AIF/MES'
                                iv_msg_number    = '085'
                                iv_msg_message_v1 = lv_ddicstruct_input
                                iv_msg_message   = lv_dummy_message
                                is_bal_context   = ls_bal_context ).
      TRY.
        lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'E' iv_do_commit = do_commit ).
       CATCH /aif/cx_enabler_base .
        MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
      ENDTRY.
      RETURN.
    ENDIF.

*   ASSIGN ls_xi_data_input->(lv_rectype_input) TO <lrs_xi_data_input>.
*   IF sy-subrc IS INITIAL.
    ASSIGN ls_xi_data_input->* TO <lrs_xi_param_input>.
*   ENDIF.
    IF NOT sy-subrc IS INITIAL.
      MESSAGE e074(/aif/mes) INTO lv_dummy_message. " where used list
      lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                iv_msg_id        = '/AIF/MES'
                                iv_msg_number    = '074'
                                iv_msg_message   = lv_dummy_message
```

```
                              is_bal_context    = ls_bal_context ).
    TRY.
      lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'E' iv_do_commit = do_commit ).
     CATCH /aif/cx_enabler_base .
      MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
    ENDTRY.
    RETURN.
  ENDIF.
 ENDIF.

* 906: create proxy object
 IF ref_to_proxy IS SUPPLIED AND ref_to_proxy IS NOT INITIAL.
  lr_proxy = ref_to_proxy.
 ELSE.
  TRY.
    CREATE OBJECT lr_proxy
     TYPE
      (ls_finf-proxyclassnamecl)
     EXPORTING
      logical_port_name = logical_port_name.
   CATCH cx_ai_system_fault INTO lx_exc.
    lv_error = 'X'.

    error_string    = lx_exc->get_text( ).
    error_long_text = lx_exc->get_longtext( ).

    MESSAGE i091(/aif/mes) INTO lv_dummy_message. "#EC * " where-used-list
    MESSAGE i000(/aif/mes) INTO lv_dummy_message. "#EC * " where-used-list

    lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                              iv_msg_id        = '/AIF/MES'
                              iv_msg_number    = '091'
                              iv_msg_message   = error_string
                              is_bal_context   = ls_bal_context
                              iv_msg_source    = /aif/if_globals=>gc_eh_msg_source-framework ).
```

```
      IF NOT error_long_text IS INITIAL.
       lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                iv_msg_id        = '/AIF/MES'
                                iv_msg_number    = '000'
                                iv_msg_message   = error_long_text
                                is_bal_context   = ls_bal_context
                                iv_msg_source    = /aif/if_globals=>gc_eh_msg_source-framework ).
      ENDIF.

      TRY.
        lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'E' iv_do_commit = do_commit ).
       CATCH /aif/cx_enabler_base .
        MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
      ENDTRY.

      MESSAGE e530(sy) RAISING general_error.
    ENDTRY.
  ENDIF.

* 1067 transform data to xi-structure (defined in customizing!)
  CALL FUNCTION '/AIF/FILE_TRANSFORM_DATA'
    EXPORTING
      ns                = ns
      ifname            = ifname
      ifversion         = ifversion
      xi_flag           = 'X'
      ifdirection       = 'O'
    IMPORTING
      out_struct        = <lrs_xi_data>
    TABLES
      return_tab        = lt_message_return
    CHANGING
      hash_tab          = lt_hash_tab
      idx_tab           = lt_idx_tab
```

```
    raw_struct           = sap_struct
    bal_context          = ls_bal_context " no need to clear before -> will be initialized within function
  EXCEPTIONS
    not_found            = 1
    customizing_incomplete = 2
    max_errors_reached    = 3
    cancel               = 4
    OTHERS               = 5.
ls_sy_old = sy.

**********************************************************************START OF
CHANGE**********************************************************************
 DATA: ls_out       TYPE edo_cl_envio_dte.
 TRY.
    ls_out = <lrs_xi_data>.
    DATA(ls_dte) = ls_out-set_dte-dte[ 1 ].
  CATCH cx_sy_itab_line_not_found.
 ENDTRY.
 ls_str = sap_struct.
 DATA(lv_comp) = ls_str-bkpf-bukrs.
 IF lv_comp IS INITIAL.
  lv_comp = ls_str-vbrk-bukrs.
 ENDIF.
*    if lv_comp IS INITIAL.
*    lv_comp = ls_str-vbrk-bukrs.
*     endif.
 DATA(lv_samp) = '123456789012'.
 SELECT * FROM t001z INTO TABLE @DATA(lt_tab) WHERE bukrs = @lv_comp.
 IF sy-subrc <> 0.
  CLEAR: lt_tab.
 ENDIF.

 READ TABLE lt_tab ASSIGNING FIELD-SYMBOL(<ls_tab>) WITH KEY party = 'TAXNR'.
 IF sy-subrc = 0.
  lv_samp = <ls_tab>-paval.
```

```
  REPLACE ALL OCCURRENCES OF '.' IN lv_samp WITH ' '.
  CONDENSE lv_samp.
  ls_out-set_dte-caratula-rut_emisor = lv_samp.
 ELSE.
  REPLACE ALL OCCURRENCES OF '.' IN ls_out-set_dte-caratula-rut_emisor WITH ' '.
  CONDENSE ls_out-set_dte-caratula-rut_emisor.
 ENDIF.



 CLEAR: lv_samp.
 READ TABLE lt_tab ASSIGNING <ls_tab> WITH KEY party = 'WT_TAX'.
 IF sy-subrc = 0.
  lv_samp = <ls_tab>-paval.
  REPLACE ALL OCCURRENCES OF '.' IN lv_samp WITH ' '.
  CONDENSE lv_samp.
  ls_out-set_dte-caratula-rut_envia = lv_samp.
 ELSE.
  REPLACE ALL OCCURRENCES OF '.' IN ls_out-set_dte-caratula-rut_envia WITH ' '.
  CONDENSE ls_out-set_dte-caratula-rut_envia.
 ENDIF.

*  REPLACE ALL OCCURRENCES OF '.' IN ls_dte-choice-documento-encabezado-emisor-rutemisor  WITH ' '.
*  ls_dte-choice-documento-encabezado-id_doc-folio = 0.
 IF ls_dte-choice-selection = 'DOCUMENTO'.
  TRY.
     IF ls_dte-choice-documento-encabezado-id_doc-tipo_dte = 56.
      ls_dte-choice-documento-referencia[ 1 ]-cod_ref   = 3.
      ls_dte-choice-documento-referencia[ 1 ]-razon_ref = 'Corrige montos'.
     ELSE.
      ls_dte-choice-documento-referencia[ 1 ]-cod_ref   = 1.
      ls_dte-choice-documento-referencia[ 1 ]-razon_ref = 'Anula Documento'.
     ENDIF.
    CATCH cx_sy_itab_line_not_found.
  ENDTRY.
  ls_dte-choice-documento-ted-dd-re   = '111'.
```

```
ls_dte-choice-documento-ted-dd-re   = '111'.
ls_dte-choice-documento-ted-dd-td   = 11.
ls_dte-choice-documento-ted-dd-f    = 11.
ls_dte-choice-documento-ted-dd-fe   = sy-datum.
ls_dte-choice-documento-ted-dd-rr   = '111'.
ls_dte-choice-documento-ted-dd-rsr  = '111'.
ls_dte-choice-documento-ted-dd-mnt  = 111.
ls_dte-choice-documento-ted-dd-it1  = '111'.
ls_dte-choice-documento-ted-dd-caf-da-re  = '111'.
ls_dte-choice-documento-ted-dd-caf-da-rs  = '111'.
ls_dte-choice-documento-ted-dd-caf-da-td  = 111.
ls_dte-choice-documento-ted-dd-caf-da-rng-d  = 111.
ls_dte-choice-documento-ted-dd-caf-da-rng-h  = 111.
ls_dte-choice-documento-ted-dd-caf-da-idk  = 111.
ls_dte-choice-documento-ted-dd-caf-da-choice-selection = 'RSAPK'.
ls_dte-choice-documento-ted-dd-caf-da-choice-rsapk-e  = '111'.
ls_dte-choice-documento-ted-dd-caf-da-choice-rsapk-m  = '111'.
ls_dte-choice-documento-ted-dd-caf-da-fa  = sy-datum.

ls_dte-choice-documento-encabezado-totales-mnt_total  = ls_dte-choice-documento-encabezado-totales-mnt_neto +
                                        ls_dte-choice-documento-encabezado-totales-iva.
IF ( ls_dte-choice-documento-encabezado-id_doc-tipo_dte = 110 OR
   ls_dte-choice-documento-encabezado-id_doc-tipo_dte = 112 ) AND ls_dte-choice-selection = 'DOCUMENTO'.
  ls_dte-choice-documento-encabezado-transporte-aduana-cod_mod_venta = 1.
  ls_dte-choice-documento-encabezado-transporte-aduana-tot_clau_venta  = ls_dte-choice-documento-encabezado-totales-mnt_total.
ENDIF.

IF ls_dte-choice-documento-encabezado-totales-iva IS INITIAL AND ls_dte-choice-documento-encabezado-id_doc-tipo_dte = 33 AND
  ( ls_dte-choice-documento-encabezado-totales-mnt_total IS NOT INITIAL ).
  ls_dte-choice-documento-encabezado-totales-mnt_exe = ls_dte-choice-documento-encabezado-totales-mnt_neto.
  CLEAR: ls_dte-choice-documento-encabezado-totales-mnt_neto.
  ls_dte-choice-documento-encabezado-id_doc-tipo_dte = 34.
  DATA(lv_dte) = ls_dte-choice-documento-encabezado-id_doc-tipo_dte.
  TRY.
     ls_dte-choice-documento-detalle[ 1 ]-ind_exe = 1.
```

```
      CATCH cx_sy_itab_line_not_found.
    ENDTRY.
  ELSEIF ls_dte-choice-documento-encabezado-totales-iva IS NOT INITIAL." AND ls_dte-choice-documento-encabezado-id_doc-tipo_dte = 33.
    TRY.
      ls_dte-choice-documento-encabezado-totales-tasa_iva = ( ( ls_dte-choice-documento-encabezado-totales-iva * 100 ) / ls_dte-choice-
documento-encabezado-totales-mnt_neto ).
      CATCH cx_sy_zerodivide.
    ENDTRY.
*     CLEAR: ls_dte-choice-documento-encabezado-totales-mnt_exe.
  ENDIF.

  TRY.
    DATA(lv_docrf) = ls_dte-choice-documento-referencia[ 1 ]-tpo_doc_ref.
    SHIFT lv_docrf LEFT DELETING LEADING '0'.
    ls_dte-choice-documento-referencia[ 1 ]-tpo_doc_ref = lv_docrf.
   CATCH cx_sy_itab_line_not_found .
  ENDTRY.
 ENDIF.
*  REPLACE ALL OCCURRENCES OF '.' IN ls_dte-choice-liquidacion-encabezado-emisor-rutemisor  WITH ' '.
*  CONDENSE ls_dte-choice-liquidacion-encabezado-emisor-rutemisor.
*  ls_dte-choice-liquidacion-encabezado-id_doc-folio = 0.

 IF ls_dte-choice-selection = 'LIQUIDACION'.
  TRY.
    IF ls_dte-choice-liquidacion-encabezado-id_doc-tipo_dte = 56.
      ls_dte-choice-liquidacion-referencia[ 1 ]-cod_ref   = 3.
      ls_dte-choice-liquidacion-referencia[ 1 ]-razon_ref = 'Corrige montos'.
    ELSE.
      ls_dte-choice-liquidacion-referencia[ 1 ]-cod_ref   = 1.
      ls_dte-choice-liquidacion-referencia[ 1 ]-razon_ref = 'Anula Documento'.
    ENDIF.
   CATCH cx_sy_itab_line_not_found.
  ENDTRY.
  ls_dte-choice-liquidacion-ted-dd-re   = '111'.
  ls_dte-choice-liquidacion-ted-dd-td   = 11.
```

```
  ls_dte-choice-liquidacion-ted-dd-f    = 11.
  ls_dte-choice-liquidacion-ted-dd-fe   = sy-datum.
  ls_dte-choice-liquidacion-ted-dd-rr   = '111'.
  ls_dte-choice-liquidacion-ted-dd-rsr  = '111'.
  ls_dte-choice-liquidacion-ted-dd-mnt  = 111.
  ls_dte-choice-liquidacion-ted-dd-it1  = '111'.
  ls_dte-choice-liquidacion-ted-dd-caf-da-re  = '111'.
  ls_dte-choice-liquidacion-ted-dd-caf-da-rs  = '111'.
  ls_dte-choice-liquidacion-ted-dd-caf-da-td  = 111.
  ls_dte-choice-liquidacion-ted-dd-caf-da-rng-d  = 111.
  ls_dte-choice-liquidacion-ted-dd-caf-da-rng-h  = 111.
  ls_dte-choice-liquidacion-ted-dd-caf-da-idk  = 111.
  ls_dte-choice-liquidacion-ted-dd-caf-da-choice-selection = 'RSAPK'.
  ls_dte-choice-liquidacion-ted-dd-caf-da-choice-rsapk-e  = '111'.
  ls_dte-choice-liquidacion-ted-dd-caf-da-choice-rsapk-m  = '111'.
  ls_dte-choice-liquidacion-ted-dd-caf-da-fa  = sy-datum.

  IF ls_dte-choice-liquidacion-encabezado-totales-iva IS INITIAL AND ls_dte-choice-liquidacion-encabezado-id_doc-tipo_dte = 33 AND
   ( ls_dte-choice-liquidacion-encabezado-totales-mnt_total IS NOT INITIAL ).
    ls_dte-choice-liquidacion-encabezado-totales-mnt_exe = ls_dte-choice-liquidacion-encabezado-totales-mnt_neto.
    CLEAR: ls_dte-choice-liquidacion-encabezado-totales-mnt_neto.
    ls_dte-choice-liquidacion-encabezado-id_doc-tipo_dte = 34.
    lv_dte = ls_dte-choice-liquidacion-encabezado-id_doc-tipo_dte.
    TRY.
       ls_dte-choice-liquidacion-detalle[ 1 ]-ind_exe = 1.
     CATCH cx_sy_itab_line_not_found.
    ENDTRY.
  ELSEIF ls_dte-choice-liquidacion-encabezado-totales-iva IS NOT INITIAL.
*     CLEAR: ls_dte-choice-liquidacion-encabezado-totales-mnt_exe.
    TRY.
       ls_dte-choice-liquidacion-encabezado-totales-tasa_iva = ( ( ls_dte-choice-liquidacion-encabezado-totales-iva * 100 ) / ls_dte-choice-
liquidacion-encabezado-totales-mnt_neto ).
      CATCH cx_sy_zerodivide.
    ENDTRY.
  ENDIF.
```

```
    TRY.
      lv_docrf = ls_dte-choice-liquidacion-referencia[ 1 ]-tpo_doc_ref.
      SHIFT lv_docrf LEFT DELETING LEADING '0'.
      ls_dte-choice-liquidacion-referencia[ 1 ]-tpo_doc_ref = lv_docrf.
     CATCH cx_sy_itab_line_not_found .
    ENDTRY.
  ENDIF.
*  REPLACE ALL OCCURRENCES OF '.' IN ls_dte-choice-exportaciones-encabezado-emisor-rutemisor  WITH ' '.
*  CONDENSE ls_dte-choice-exportaciones-encabezado-emisor-rutemisor.
*  ls_dte-choice-exportaciones-encabezado-id_doc-folio = 0.
  IF ls_dte-choice-selection = 'EXPORTACIONES'.
    TRY.
      IF ls_dte-choice-exportaciones-encabezado-id_doc-tipo_dte = 56.
        ls_dte-choice-exportaciones-referencia[ 1 ]-cod_ref   = 3.
        ls_dte-choice-exportaciones-referencia[ 1 ]-razon_ref = 'Corrige montos'.
      ELSE.
        ls_dte-choice-exportaciones-referencia[ 1 ]-cod_ref   = 1.
        ls_dte-choice-exportaciones-referencia[ 1 ]-razon_ref = 'Anula Documento'.
      ENDIF.
     CATCH cx_sy_itab_line_not_found.
    ENDTRY.
    ls_dte-choice-exportaciones-ted-dd-re   = '111'.
    ls_dte-choice-exportaciones-ted-dd-td   = 11.
    ls_dte-choice-exportaciones-ted-dd-f    = 11.
    ls_dte-choice-exportaciones-ted-dd-fe   = sy-datum.
    ls_dte-choice-exportaciones-ted-dd-rr   = '111'.
    ls_dte-choice-exportaciones-ted-dd-rsr  = '111'.
    ls_dte-choice-exportaciones-ted-dd-mnt  = 111.
    ls_dte-choice-exportaciones-ted-dd-it1  = '111'.
    ls_dte-choice-exportaciones-ted-dd-caf-da-re  = '111'.
    ls_dte-choice-exportaciones-ted-dd-caf-da-rs  = '111'.
    ls_dte-choice-exportaciones-ted-dd-caf-da-td  = 111.
    ls_dte-choice-exportaciones-ted-dd-caf-da-rng-d  = 111.
    ls_dte-choice-exportaciones-ted-dd-caf-da-rng-h  = 111.
```

```
  ls_dte-choice-exportaciones-ted-dd-caf-da-idk  = 111.
  ls_dte-choice-exportaciones-ted-dd-caf-da-choice-selection = 'RSAPK'.
  ls_dte-choice-exportaciones-ted-dd-caf-da-choice-rsapk-e  = '111'.
  ls_dte-choice-exportaciones-ted-dd-caf-da-choice-rsapk-m  = '111'.
  ls_dte-choice-exportaciones-ted-dd-caf-da-fa  = sy-datum.
  IF ( ls_dte-choice-exportaciones-encabezado-id_doc-tipo_dte = 110 OR
    ls_dte-choice-exportaciones-encabezado-id_doc-tipo_dte = 112 ) AND ls_dte-choice-selection = 'EXPORTACIONES'.
    ls_dte-choice-exportaciones-encabezado-transporte-aduana-cod_mod_venta = 1.
    ls_dte-choice-exportaciones-encabezado-transporte-aduana-tot_clau_venta  = ls_dte-choice-exportaciones-encabezado-totales-mnt_total.
  ENDIF.

  TRY.
      lv_docrf = ls_dte-choice-exportaciones-referencia[ 1 ]-tpo_doc_ref.
      SHIFT lv_docrf LEFT DELETING LEADING '0'.
      ls_dte-choice-exportaciones-referencia[ 1 ]-tpo_doc_ref = lv_docrf.
    CATCH cx_sy_itab_line_not_found .
  ENDTRY.
 ENDIF.
*********************************************************************
**Change by Kriti -> value of GIRO_EMIS not populated fix:
 READ TABLE lt_tab ASSIGNING <ls_tab> WITH KEY party = 'GIROEM'.
 IF sy-subrc = 0.
  ls_dte-choice-documento-encabezado-emisor-giro_emis     = <ls_tab>-paval.
  ls_dte-choice-exportaciones-encabezado-emisor-giro_emis = <ls_tab>-paval.
  ls_dte-choice-liquidacion-encabezado-emisor-giro_emis   = <ls_tab>-paval.
 ENDIF.
 ls_dte-choice-documento-encabezado-emisor-rutemisor     = ls_out-set_dte-caratula-rut_emisor.
 ls_dte-choice-liquidacion-encabezado-emisor-rutemisor   = ls_out-set_dte-caratula-rut_emisor.
 ls_dte-choice-exportaciones-encabezado-emisor-rutemisor = ls_out-set_dte-caratula-rut_emisor.
*********************************************************************

 TRY.
    ls_out-set_dte-dte[ 1 ] = ls_dte.
    IF lv_dte = 34.
      TRY.
```

```
          ls_out-set_dte-caratula-sub_tot_dte[ 1 ]-tpo_dte = lv_dte.

        CATCH cx_sy_itab_line_not_found.

       ENDTRY.

     ENDIF.

     <lrs_xi_data> = ls_out.

   CATCH cx_sy_itab_line_not_found.

 ENDTRY.

**********************************************************************END OF
CHANGE**********************************************************************

  CALL METHOD lr_enabler->add_bapiret2_messages(

     it_bapiret2_messages = lt_message_return

     is_bal_context       = ls_bal_context

     iv_msg_source        = /aif/if_globals=>gc_eh_msg_source-interface

   ).

  CLEAR lt_message_return.


  lt_all_bal_messages = lr_enabler->get_log_messages( iv_include_buffered = 'X' ).


  IF NOT ls_sy_old-subrc IS INITIAL AND NOT ls_sy_old-msgty IS INITIAL.

    " Make sure the framework error message is included in log - add if not

    READ TABLE lt_all_bal_messages WITH KEY msgty = ls_sy_old-msgty msgid = ls_sy_old-msgid msgno = ls_sy_old-msgno TRANSPORTING NO
FIELDS.

    IF NOT sy-subrc IS INITIAL.

      lr_enabler->add_single_log_message( iv_msg_type      = ls_sy_old-msgty

                                iv_msg_id        = ls_sy_old-msgid

                                iv_msg_number    = ls_sy_old-msgno

                                iv_msg_message_v1 = ls_sy_old-msgv1

                                iv_msg_message_v2 = ls_sy_old-msgv2

                                iv_msg_message_v3 = ls_sy_old-msgv3

                                iv_msg_message_v4 = ls_sy_old-msgv4

                                is_bal_context    = ls_bal_context

                                iv_msg_source     = /aif/if_globals=>gc_eh_msg_source-framework ).

      " lt_all_bal_messages needs to be updated now, since a new message was added

      lt_all_bal_messages = lr_enabler->get_log_messages( iv_include_buffered = 'X' ).

    ENDIF.

  ENDIF.
```

```
* note 2148556
 lv_current_msg_state = 'I'.
 READ TABLE lt_all_bal_messages WITH KEY msgty = 'A' TRANSPORTING NO FIELDS.
 IF sy-subrc IS INITIAL.
  lv_current_msg_state = 'A'.
 ELSE.
  READ TABLE lt_all_bal_messages WITH KEY msgty = 'E' TRANSPORTING NO FIELDS.
  IF sy-subrc IS INITIAL.
   lv_current_msg_state = 'E'.
  ENDIF.
 ENDIF.
* note 2148556

 TRY.
    " Keyfields need to be updated after transform - so provide raw + sap structure
    lr_enabler->/aif/if_enabler_base~update(
     iv_message_status_flag     = lv_current_msg_state
     is_sap_structure           = sap_struct
     is_raw_structure           = <lrs_xi_data>
     iv_do_commit               = do_commit
    ).
  CATCH /aif/cx_enabler_base .
   MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
 ENDTRY.

 IF lv_current_msg_state = 'E' OR lv_current_msg_state = 'A'.
  MESSAGE e107(/aif/mes) RAISING transformation_error.
 ENDIF.

 MESSAGE i082(/aif/mes) INTO lv_dummy_message. " where used list
 lr_enabler->add_single_log_message( iv_msg_type      = 'I'
                                     iv_msg_id        = '/AIF/MES'
                                     iv_msg_number    = '082'
                                     iv_msg_message   = lv_dummy_message
```

```
                              is_bal_context    = ls_bal_context ).
  TRY.
     lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'I' iv_do_commit = do_commit ).
    CATCH /aif/cx_enabler_base .
     MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
  ENDTRY.

* 1314ff call method of proxy object
  CLEAR lv_error.

  TRY.
* determine all parameters for the mehtod from SPROXDAT
     CLEAR lt_sproxdat.
     SELECT * FROM sproxdat INTO ls_sproxdat
       WHERE object = sprx_const_class AND obj_name = ls_finf-proxyclassnamecl
       AND object1 = sprx_const_method AND obj_name1 = lv_methodname.
       APPEND ls_sproxdat TO lt_sproxdat.
     ENDSELECT.
     REFRESH lt_parmbind.
     IF lv_sync IS INITIAL.
*specify queue id
       IF iv_eoio IS NOT INITIAL AND iv_queue_id IS NOT INITIAL.
         CALL METHOD lr_proxy->('IF_PROXY_BASIS~GET_PROTOCOL')
           EXPORTING
             protocol_name = if_wsprotocol=>async_messaging
           RECEIVING
             protocol      = lr_protocol.
         lr_async_messaging ?= lr_protocol.
         IF lr_async_messaging IS BOUND.
           lr_async_messaging->set_serialization_context( iv_queue_id ).
         ENDIF.
       ENDIF.
*      fill parmbind table depending on sproxdat with type PAIM, importing
       LOOP AT lt_sproxdat ASSIGNING <fs_sproxdat>.
         CLEAR ls_parmbind_line.
```

```
        CASE <fs_sproxdat>-object2.
          WHEN sprx_const_param_importing.
            ls_parmbind_line-kind = cl_abap_objectdescr=>exporting.
            GET REFERENCE OF <lrs_xi_param> INTO ls_parmbind_line-value.
          WHEN OTHERS.
            CONTINUE.
        ENDCASE.
        ls_parmbind_line-name = <fs_sproxdat>-obj_name2.
        INSERT ls_parmbind_line INTO TABLE lt_parmbind.
      ENDLOOP.
      CALL METHOD lr_proxy->(lv_methodname)
        PARAMETER-TABLE
        lt_parmbind.
    ELSE.
*     fill parmbind table depending on sproxdat with type PAIM and PAEM
      LOOP AT lt_sproxdat ASSIGNING <fs_sproxdat>.
        CLEAR ls_parmbind_line.
        CASE <fs_sproxdat>-object2.
          WHEN sprx_const_param_exporting.
            ls_parmbind_line-kind = cl_abap_objectdescr=>importing.
            GET REFERENCE OF <lrs_xi_param_input> INTO ls_parmbind_line-value.
          WHEN sprx_const_param_importing.
            ls_parmbind_line-kind = cl_abap_objectdescr=>exporting.
            GET REFERENCE OF <lrs_xi_param> INTO ls_parmbind_line-value.
          WHEN OTHERS.
            CONTINUE.
        ENDCASE.
        ls_parmbind_line-name = <fs_sproxdat>-obj_name2.
        INSERT ls_parmbind_line INTO TABLE lt_parmbind.
      ENDLOOP.
* call mehtod with generic parameterlist determined from sproxdat
      CALL METHOD lr_proxy->(lv_methodname)
        PARAMETER-TABLE
        lt_parmbind.
    ENDIF.
```

```
    /aif/cl_proxy_outbound=>clear_current_tmp_msgguid( ).

  CATCH cx_ai_system_fault cx_ai_application_fault INTO lx_exc.
    lv_error       = 'X'.

    /aif/cl_proxy_outbound=>clear_current_tmp_msgguid( ).

* Check if a fault handler is defined for the interface or not
    IF ls_finf-fault_handler IS INITIAL.
      error_string    = lx_exc->get_text( ).
      error_long_text = lx_exc->get_longtext( ).
    ELSE.
      TRY.
        CALL FUNCTION ls_finf-fault_handler
          EXPORTING
            ir_fault          = lx_exc
          IMPORTING
            ev_error_string    = error_string
            ev_error_long_text = error_long_text
            et_bapireturn      = lt_fault.
       CATCH cx_root.
        error_string    = lx_exc->get_text( ).
        error_long_text = lx_exc->get_longtext( ).
      ENDTRY.
    ENDIF.

    IF 1 = 2.
      MESSAGE i091(/aif/mes) WITH '' '' '' '' INTO lv_dummy_message. " where-used-list
      MESSAGE i000(/aif/mes) WITH '' '' '' '' INTO lv_dummy_message. " where-used-list
    ENDIF.

    lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                        iv_msg_id        = '/AIF/MES'
                                        iv_msg_number    = '091'
                                        iv_msg_message   = error_string
```

```
                              is_bal_context    = ls_bal_context
                              iv_msg_source     = /aif/if_globals=>gc_eh_msg_source-framework ).

    IF NOT error_long_text IS INITIAL.
     lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                iv_msg_id        = '/AIF/MES'
                                iv_msg_number    = '000'
                                iv_msg_message   = error_long_text
                                is_bal_context   = ls_bal_context
                                iv_msg_source    = /aif/if_globals=>gc_eh_msg_source-framework ).
    ENDIF.
    IF lt_fault IS INITIAL AND lx_exc IS INSTANCE OF cx_ai_application_fault.
      "try to collect error information from application specific exception object
*       PERFORM get_exception_messages USING lx_exc lr_enabler.
    ENDIF.
    IF NOT lt_fault IS INITIAL.
     lr_enabler->add_bapiret2_messages( it_bapiret2_messages = lt_fault
                              is_bal_context      = ls_bal_context
                              iv_msg_source       = /aif/if_globals=>gc_eh_msg_source-framework ).
    ENDIF.

    TRY.
       lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'I' iv_do_commit = do_commit ).
     CATCH /aif/cx_enabler_base .
       MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
    ENDTRY.
  ENDTRY.

**********************************************************************START OF
CHANGE********************************************************************************


  FIELD-SYMBOLS: <ls_resp> TYPE any,
                 <ls_out>  TYPE any.
  DATA: ls_resp         TYPE edo_cl_envio_dteresponse,
        ls_req          TYPE edo_cl_envio_dte1,
```

```
      ls_edocumentfile TYPE zfid_edoc_pdf_cl.
  READ TABLE lt_parmbind ASSIGNING FIELD-SYMBOL(<ls_parm>) WITH KEY name = 'INPUT'.
  IF sy-subrc = 0.
    ASSIGN <ls_parm>-value->* TO <ls_resp>."De-reference the data reference
*   IF <ls_resp> IS BOUND.
    ls_resp = <ls_resp>.
*   ENDIF.
  ENDIF.
  READ TABLE lt_parmbind ASSIGNING <ls_parm> WITH KEY name = 'OUTPUT'.
  IF sy-subrc = 0.
    ASSIGN <ls_parm>-value->* TO <ls_out>."De-reference the data reference
*   IF <ls_out> IS BOUND.
    ls_req = <ls_out>.
*   ENDIF.
  ENDIF.
  IF    ls_resp-envio_dteresponse-signed_dte IS NOT INITIAL OR ls_resp-envio_dteresponse-digital_seal IS NOT INITIAL.

    SELECT * FROM zfid_edoc_pdf_cl INTO TABLE @DATA(lt_efile) WHERE edoc_guid = @ls_req-envio_dte-set_dte-id.
    IF sy-subrc = 0.
      SORT lt_efile BY seq_no DESCENDING.
      READ TABLE lt_efile ASSIGNING FIELD-SYMBOL(<ls_efile>) INDEX 1.
      IF sy-subrc = 0.
        ls_edocumentfile-seq_no = <ls_efile>-seq_no + 1.
      ENDIF.
    ELSE.
      ls_edocumentfile-seq_no = 1.
    ENDIF.

    DATA(ls_tmp) = ls_resp-envio_dteresponse-response_dte.
    TRANSLATE ls_tmp TO UPPER CASE.
    SPLIT ls_tmp AT '<FOLIO>' INTO DATA(ls_tmp1) DATA(ls_tmp2).
    SPLIT ls_tmp2 AT '</FOLIO>' INTO DATA(lv_folio) ls_tmp1.

    ls_tmp = ls_resp-envio_dteresponse-response_dte.
    TRANSLATE ls_tmp TO UPPER CASE.
```

```
  SPLIT ls_tmp AT '<TIPODTE>' INTO ls_tmp1 ls_tmp2.
  SPLIT ls_tmp2 AT '</TIPODTE>' INTO DATA(lv_dte_typ) ls_tmp1.

  ls_edocumentfile-dte_type   = CONV #( lv_dte_typ ).
  ls_edocumentfile-dte_number = CONV #( lv_folio ).
****************************** PDF related ***********************
  ls_edocumentfile-file_guid = cl_edoc_util=>create_uuid( ).
  ls_edocumentfile-edoc_guid = ls_req-envio_dte-set_dte-id.
  ls_edocumentfile-create_date = sy-datum.
  ls_edocumentfile-create_time = sy-uzeit.

*    File name and file content
  CONCATENATE 'CL_' sy-datum '_' sy-uzeit '.pdf'         "#EC NOTEXT
       INTO ls_edocumentfile-file_name.

  ls_edocumentfile-file_raw    = ls_resp-envio_dteresponse-signed_dte.

  DATA: lv_success TYPE boolean,
      lv_subobj  TYPE nrsobj.

  SELECT SINGLE bukrs FROM edocument INTO @DATA(lv_bukrs) WHERE edoc_guid = @ls_edocumentfile-edoc_guid.
  IF sy-subrc = 0.
   lv_subobj = lv_bukrs.
  ENDIF.

  CALL FUNCTION 'Z_FI_EDOC_CHILE_PDF_ENTRY' DESTINATION 'NONE'
    EXPORTING
     ls_entry = ls_edocumentfile.
****************************** PDF related ***********************

  CALL FUNCTION 'Z_FI_EDOC_CHILE_UPD_NO_RANGE' DESTINATION 'NONE'
    EXPORTING
     iv_dte_number = ls_edocumentfile-dte_number
     iv_subobj     = lv_subobj
     iv_dte_type   = ls_edocumentfile-dte_type
```

```
  IMPORTING
    ev_success    = lv_success.


ELSE.
 TRY.
    IF ls_req-envio_dte-set_dte-dte[ 1 ]-choice-selection = 'DOCUMENTO'.
     DATA: lv_edoc     TYPE edoc_guid,
          lv_folio_num TYPE edoc_cl_number,
          lv_tipo_dte  TYPE edoc_cl_dte.
     lv_edoc = ls_req-envio_dte-set_dte-id.
     SELECT SINGLE bukrs FROM edocument INTO @lv_bukrs WHERE edoc_guid = @lv_edoc.
     IF sy-subrc = 0.
      lv_subobj = lv_bukrs.
     ENDIF.
     lv_tipo_dte  = ls_req-envio_dte-set_dte-dte[ 1 ]-choice-documento-encabezado-id_doc-tipo_dte.
     IF strlen( lv_tipo_dte ) = 2.
      lv_tipo_dte = '0' && lv_tipo_dte.
     ENDIF.
     lv_folio_num = ls_req-envio_dte-set_dte-dte[ 1 ]-choice-documento-encabezado-id_doc-folio - 1.
    ELSEIF ls_req-envio_dte-set_dte-dte[ 1 ]-choice-selection = 'LIQUIDACION'.
     lv_edoc = ls_req-envio_dte-set_dte-id.
     SELECT SINGLE bukrs FROM edocument INTO @lv_bukrs WHERE edoc_guid = @lv_edoc.
     IF sy-subrc = 0.
      lv_subobj = lv_bukrs.
     ENDIF.
     lv_tipo_dte  = ls_req-envio_dte-set_dte-dte[ 1 ]-choice-liquidacion-encabezado-id_doc-tipo_dte.
     IF strlen( lv_tipo_dte ) = 2.
      lv_tipo_dte = '0' && lv_tipo_dte.
     ENDIF.
     lv_folio_num = ls_req-envio_dte-set_dte-dte[ 1 ]-choice-liquidacion-encabezado-id_doc-folio - 1.
    ELSE.
     lv_edoc = ls_req-envio_dte-set_dte-id.
     SELECT SINGLE bukrs FROM edocument INTO @lv_bukrs WHERE edoc_guid = @lv_edoc.
     IF sy-subrc = 0.
      lv_subobj = lv_bukrs.
```

```
        ENDIF.
        lv_tipo_dte  = ls_req-envio_dte-set_dte-dte[ 1 ]-choice-exportaciones-encabezado-id_doc-tipo_dte.
        IF strlen( lv_tipo_dte ) = 2.
          lv_tipo_dte = '0' && lv_tipo_dte.
        ENDIF.
        lv_folio_num = ls_req-envio_dte-set_dte-dte[ 1 ]-choice-exportaciones-encabezado-id_doc-folio - 1.

      ENDIF.
    CATCH cx_sy_itab_line_not_found.
   ENDTRY.
   CALL FUNCTION 'Z_FI_EDOC_CHILE_UPD_NO_RANGE' DESTINATION 'NONE'
     EXPORTING
       iv_dte_number = lv_folio_num
       iv_subobj     = lv_subobj
       iv_dte_type   = lv_tipo_dte
     IMPORTING
       ev_success    = lv_success.
 ENDIF.
**********************************************************************END OF
CHANGE**********************************************************************************

* determination of inbound interface that is for processing response message
  IF lv_error IS INITIAL AND NOT lv_sync IS INITIAL.
    CLEAR ls_inbound_finf.
    CALL FUNCTION '/AIF/DETERMINE_INTERFACE_QD'
      EXPORTING
        input        = <lrs_xi_param_input>
        iv_proxyclass = ls_finf-proxyclassnamecl
        iv_prx_method = ls_finf-prx_method
        iv_sync      = 'X'
      IMPORTING
        finf         = ls_inbound_finf
      EXCEPTIONS
        error        = 1
        OTHERS       = 2.
```

```
  IF sy-subrc <> 0 OR ls_inbound_finf IS INITIAL.
*     in general, the response message shall be an xml engine interface.
    DATA: lr_if_det_engine_xml TYPE REF TO /aif/cl_inf_det_engine_xml,
          lv_name              TYPE /aif/lfieldname_infdet.
    TRY.
      lr_if_det_engine_xml ?= /aif/cl_aif_engine_factory=>get_inf_det_engine(
                                  iv_type          = /aif/cl_pers_config=>c_if_type_xml
                                  iv_cust_ns       = space
                                  iv_cust_type     = space
                                  ).
      CALL METHOD lr_if_det_engine_xml->get_typename
       EXPORTING
        iv_input = <lrs_xi_param_input>
       IMPORTING
        ev_name1 = lv_name.

      CALL METHOD lr_if_det_engine_xml->/aif/if_inf_det_engine~determine_inf
       EXPORTING
        iv_name1   = lv_name
        iv_name2   = space
        iv_input   = <lrs_xi_param_input>
        iv_msgguid = space
       IMPORTING
        es_finf    = ls_inbound_finf.

     CATCH /aif/cx_aif_engine_not_found
         /aif/cx_error_handling_general
         /aif/cx_inf_det_base
         /aif/cx_aif_engine_base.
      lv_error = 'X'.
    ENDTRY.

    IF ls_inbound_finf IS INITIAL.
     IF 1 = 2. MESSAGE e004(/aif/engine_xml) WITH lv_name. ENDIF .
     lr_enabler->add_single_log_message( iv_msg_type      = 'E'
```

```
                                iv_msg_id        = '/AIF/ENGINE_XML'
                                iv_msg_number    = '004'
                                iv_msg_message_v1 = lv_name
                               is_bal_context    = ls_bal_context ).
      TRY.
        lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'E' iv_do_commit = do_commit ).
       CATCH /aif/cx_enabler_base .
        MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
      ENDTRY.
      MESSAGE e004(/aif/engine_xml) WITH lv_name
           RAISING general_error.
    ENDIF.
  ENDIF.
 ENDIF.

* 1374ff perform actions
 IF lv_error IS INITIAL.
   " Note that lt_intrec and lt_recflds are not used in FILE_PERFORM_ACTIONS - so we do not provide them here
   " Note that lt_return is not supplied here as well, as it only yields as source for messages to be appended
   "   to RETURN_TAB by FILE_PERFORM_ACTIONS. lt_return was always transferred empty - so no need to use it here

*<<<Note 1877297
* In Assign Action, no specific Record Type is defined.
* In such case, the action is not invoked
* this is due to importing parameter OUT_DDIC_TREE of FM FILE_PERFORM_ACTION is not used.
   CALL FUNCTION '/AIF/GET_DEST_STRUCT'
     EXPORTING
       is_finf    = ls_finf
     CHANGING
       cv_tabname = lv_tabname
     EXCEPTIONS
       error      = 1
       OTHERS     = 2.
   IF sy-subrc <> 0.
* don't find specific Exception
```

```
* so, just add a log message
* FM will quit later due to an error which is related to this failure.
    lr_enabler->add_single_log_message( iv_msg_type      = sy-msgty
                                iv_msg_id        = sy-msgid
                                iv_msg_number    = sy-msgno
*                                 iv_msg_message   = error_long_text
                                is_bal_context   = ls_bal_context
                                iv_msg_source    = /aif/if_globals=>gc_eh_msg_source-framework ).
  ENDIF.



  CALL FUNCTION '/AIF/UTIL_GET_STRUCT_FIELDS'
    EXPORTING
      structname = lv_tabname
    IMPORTING
      ddic_tree  = lt_out_ddic_tree
    EXCEPTIONS
      error     = 1
      cancel    = 2
      OTHERS    = 3.
  IF sy-subrc <> 0.
* don't find specific Exception
* so, just add a log message
* FM will quit later due to an error which is related to this failure.
    lr_enabler->add_single_log_message( iv_msg_type      = sy-msgty
                               iv_msg_id        = sy-msgid
                               iv_msg_number    = sy-msgno
*                                 iv_msg_message   = error_long_text
                               is_bal_context   = ls_bal_context
                               iv_msg_source    = /aif/if_globals=>gc_eh_msg_source-framework ).


  ENDIF.
*<<<Note 1877297


*<<<Begin of Note 2809552
```

```
  TRY.
      CALL METHOD lr_proxy->('GET_PROTOCOL')
       EXPORTING
         protocol_name = 'IF_WSPROTOCOL_MESSAGE_ID'
       RECEIVING
         protocol      = lr_prot.
      lr_msgid_prot    ?= lr_prot.
      ximsgguid        = lr_msgid_prot->get_message_id( ).
      lv_ximsgguid_cast = ximsgguid.
      ximsgguid_out     = ximsgguid.
*     gv_msgguid       = ximsgguid.
      IF msgguid IS INITIAL.
*     have to dequeue old msgguid here, since further process do not have old msgguid info
       CALL FUNCTION 'DEQUEUE_/AIF/EMSGGUID'
         EXPORTING
           mode_/aif/msgguid_st = 'E'
           mandt              = lv_mandt
           msgguid            = lv_guid_32_temp
           _scope             = 1
         EXCEPTIONS
           error_message      = 1
           OTHERS             = 2.
       IF sy-subrc <> 0.
         MESSAGE ID sy-msgid TYPE 'S' NUMBER sy-msgno
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
       ENDIF.
      ENDIF.
      CLEAR lv_msg_locked.
      lv_mandt = cl_abap_syst=>get_client( ).
      DO 5 TIMES.
       CALL FUNCTION 'ENQUEUE_/AIF/EMSGGUID'
         EXPORTING
           mode_/aif/msgguid_st = 'E'
           mandt              = lv_mandt
           _scope             = '1' "in program scope
```

```
      msgguid             = lv_ximsgguid_cast
    EXCEPTIONS
      foreign_lock        = 1
      system_failure      = 2
      OTHERS              = 3.
   IF sy-subrc <> 0.
    lv_msg_locked = abap_false.
    WAIT UP TO 1 SECONDS.
   ELSE.
    lv_msg_locked = abap_true.
    EXIT.
   ENDIF.
   IF lv_msg_locked = abap_false.
    MESSAGE e028(/aif/runtime) WITH lv_guid_32_temp RAISING general_error.
   ENDIF.
  ENDDO.
 CATCH cx_ai_system_fault cx_ai_application_fault INTO lx_exc.
  error_string    = lx_exc->get_text( ).
  error_long_text = lx_exc->get_longtext( ).

  IF 1 = 2.
   MESSAGE e091(/aif/mes) WITH '' '' '' '' INTO lv_dummy_message. " where-used-list
   MESSAGE e000(/aif/mes) WITH '' '' '' '' INTO lv_dummy_message. " where-used-list
  ENDIF.

  lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                      iv_msg_id        = '/AIF/MES'
                                      iv_msg_number    = '091'
                                      iv_msg_message_v1 = 'GET_PROTOCOL'
                                      iv_msg_message   = error_string
                                      is_bal_context   = ls_bal_context
                                      iv_msg_source    = /aif/if_globals=>gc_eh_msg_source-framework ).

  IF NOT error_long_text IS INITIAL.
   lr_enabler->add_single_log_message( iv_msg_type      = 'E'
```

```
                                iv_msg_id        = '/AIF/MES'
                                iv_msg_number    = '000'
                                iv_msg_message   = error_long_text
                                is_bal_context   = ls_bal_context
                                iv_msg_source    = /aif/if_globals=>gc_eh_msg_source-framework ).
      ENDIF.
      TRY.
        lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'I' iv_do_commit = do_commit ).
       CATCH /aif/cx_enabler_base .
        IF msgguid IS INITIAL.
*      have to dequeue old msgguid here, since further process do not have old msgguid
          CALL FUNCTION 'DEQUEUE_/AIF/EMSGGUID'
           EXPORTING
            mode_/aif/msgguid_st = 'E'
            mandt              = lv_mandt
            msgguid            = lv_guid_32_temp
            _scope             = 1
           EXCEPTIONS
            error_message      = 1
            OTHERS             = 2.
          IF sy-subrc <> 0.
           MESSAGE ID sy-msgid TYPE 'S' NUMBER sy-msgno
             WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
          ENDIF.
        ENDIF.
        MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
      ENDTRY.
    ENDTRY.
*<<<End of Note 2809552

    CALL FUNCTION '/AIF/FILE_PERFORM_ACTIONS'
     EXPORTING
      ls_finf              = ls_finf
      testrun              = space
      lt_log_handle_save   = lt_log_handle_save
```

```
    lv_log_handle         = '' " initial because we take care of writing messages to bal log ourselves
    out_ddic_tree         = lt_out_ddic_tree "Note 1877297
  TABLES
    lt_acfields           = lt_acfields
    lt_accheck            = lt_accheck
    lt_check              = lt_check
    lt_tabchk             = lt_tabchk
    lt_tabfld             = lt_tabfld
    lt_ifacts             = lt_ifacts
    lt_actions            = lt_actions
    lt_actionts           = lt_actionts
    lt_funcs              = lt_funcs
    lt_intrec             = lt_intrec
    lt_recflds            = lt_recflds
    lt_return             = lt_temp_return
    return_tab            = lt_message_return  " will include new messages raised from FILE_PERFORM_ACTIONS
  CHANGING
    ls_bal_context        = ls_bal_context
    data                  = <lrs_xi_data>  " TODO: clarify if lrs_xi_data or ls_sap_scruct needs to be provided here
    hash_tab              = lt_hash_tab
    idx_tab               = lt_idx_tab
    lv_error_all          = lv_error_all
    successflag           = lv_successflag
  EXCEPTIONS
    customizing_incomplete = 1        " successflag = lv_successflag " never used
    cancel                 = 2
    OTHERS                 = 3.
  IF NOT sy-subrc IS INITIAL OR NOT lv_error_all IS INITIAL.
    ROLLBACK WORK.                              "#EC CI_ROLLBACK
    lv_error = 'X'.
    lv_error_all = 'X'.
  ELSEIF do_commit = 'X'.
    COMMIT WORK AND WAIT.
  ENDIF.
ENDIF.
```

```
* /AIF/FILE_PERFORM_ACTIONS will clear ls_bal_context-msg_category
 ls_bal_context-msg_category = 'O'.
 ls_bal_context-ifaction     = appl_log_function.

 " TODO: BAL_CONTEXT does not work here: bal context is changed within FILE_PERFORM_ACTIONS
 "      and a different bal context is used for different messages -> however, we have no
 "      possibility to receive the bal context for the bapiret2 messages.
 "
 "      Possible solution: we need a temporary bal log and hand over the lognumber to
 "                    FILE_PERFORM_ACTIONS -> later read messages from this log
 CALL METHOD lr_enabler->add_bapiret2_messages(
    it_bapiret2_messages = lt_message_return
    is_bal_context       = ls_bal_context
    iv_msg_source        = /aif/if_globals=>gc_eh_msg_source-interface
  ).
 CLEAR lt_message_return.

 ls_bal_context-ifaction = 'AIF_SENDING'.
 ls_bal_context-actionnr = '001'.
 ls_bal_context-sydate   = sy-datum.
 ls_bal_context-sytime   = sy-uzeit.
 MESSAGE i080(/aif/mes) INTO lv_dummy_message. " where-used-list
 lr_enabler->add_single_log_message( iv_msg_type      = 'I'
                             iv_msg_id        = '/AIF/MES'
                             iv_msg_number    = '080'
                             iv_msg_message   = lv_dummy_message
                             is_bal_context   = ls_bal_context ).
 TRY.
    " UPDATE KEYFIELDS NECESSARY, since values might have been changed in actions called above
    lr_enabler->/aif/if_enabler_base~update(
      iv_message_status_flag = 'I'
      is_sap_structure          = sap_struct
      is_raw_structure          = <lrs_xi_data>
      iv_do_commit = do_commit ).
  CATCH /aif/cx_enabler_base .
```

```
    MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
  ENDTRY.

  IF lv_error_all = 'X'. "Note 289367
    lv_current_message_location = /aif/if_globals=>gc_message_location-action.
  ELSE.
    lv_current_message_location = /aif/if_globals=>gc_message_location-proxy_fwk.
  ENDIF.

* 1468ff. get messageID by protocol of the proxy object
  TRY.
*Note 2151693
* a typical use case of the BADI method is to replace a temporary generated outbound message GUID
* with the message GUID used by the proxy framework in customer specific logic
      GET BADI lr_runtime_badi.
      IF lr_runtime_badi IS BOUND.
        lv_guid_32 = ximsgguid.
        CALL BADI lr_runtime_badi->transfer_message_guid
          EXPORTING
            iv_ns              = ns
            iv_ifname          = ifname
            iv_ifversion       = ifversion
            iv_temp_request_guid = lv_guid_32_temp
            iv_request_guid    = lv_guid_32.
      ENDIF.

      CLEAR lr_prot.
      CALL METHOD lr_proxy->('GET_PROTOCOL')
        EXPORTING
          protocol_name = 'INTERNAL'
        RECEIVING
          protocol      = lr_prot.
      lr_msgid_internal ?= lr_prot.
      lr_xmb ?= lr_msgid_internal->get_xi_message_object( ).
      IF lr_xmb IS NOT INITIAL.
```

```
      lr_runtime = lr_xmb->get_msghdr30_runtime( ).
      lv_pipelineid_ext = lr_runtime->get_external_pl_id( ).
      lv_pipelineid = lv_pipelineid_ext.
    ELSE.
      lv_pipelineid = 'WS_SENDER'.
    ENDIF.

**********Note 2007548 Set outbound PID*************

    lr_enabler->set_pid( lv_pipelineid ).

**********Note 2007548 Set outbound PID*************
    TRY.
       lr_enabler->/aif/if_enabler_base~update(
         iv_message_status_flag     = 'I'
         iv_msgguid                 = lv_ximsgguid_cast
         iv_message_aif_location    = lv_current_message_location "Note 289367
         iv_do_commit               = ''
       ).
      CATCH /aif/cx_enabler_base .
       MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
    ENDTRY.
   CATCH cx_ai_system_fault cx_ai_application_fault INTO lx_exc.
    error_string    = lx_exc->get_text( ).
    error_long_text = lx_exc->get_longtext( ).

    IF 1 = 2.
      MESSAGE e091(/aif/mes) WITH '' '' '' '' INTO lv_dummy_message. " where-used-list
      MESSAGE e000(/aif/mes) WITH '' '' '' '' INTO lv_dummy_message. " where-used-list
    ENDIF.

    lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                iv_msg_id        = '/AIF/MES'
                                iv_msg_number    = '091'
                                iv_msg_message_v1 = 'GET_PROTOCOL'
```

```
                              iv_msg_message    = error_string
                              is_bal_context    = ls_bal_context
                              iv_msg_source     = /aif/if_globals=>gc_eh_msg_source-framework ).

    IF NOT error_long_text IS INITIAL.
     lr_enabler->add_single_log_message( iv_msg_type      = 'E'
                                iv_msg_id        = '/AIF/MES'
                                iv_msg_number    = '000'
                                iv_msg_message   = error_long_text
                                is_bal_context   = ls_bal_context
                                iv_msg_source    = /aif/if_globals=>gc_eh_msg_source-framework ).
    ENDIF.
    TRY.
       lr_enabler->/aif/if_enabler_base~update( iv_message_status_flag = 'I' iv_do_commit = do_commit ).
     CATCH /aif/cx_enabler_base .
       MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
    ENDTRY.
  ENDTRY.

  IF lv_error IS INITIAL AND NOT lv_sync IS INITIAL.
   DATA: lv_msgguid_sender TYPE sxmsmguid.
   lv_msgguid_sender = ximsgguid.
   SELECT SINGLE msgguid INTO ximsgguid
    FROM sxmspmast
    WHERE ref_to_msg = lv_msgguid_sender
     AND pid       = lv_pipelineid.
   IF sy-subrc <> 0.
*   Note 2144871
*   in case synchronous service calls are not logged in the local integration engine
*   the select will not return a result. Create a new GUID in that case
    TRY.
       lv_new_guid = cl_system_uuid=>create_uuid_c32_static( ).
     CATCH cx_uuid_error.
    ENDTRY.
    ximsgguid = lv_new_guid.
```

```
  ELSE.
    lv_new_guid = ximsgguid.
  ENDIF.
*Note 2151693: a typical use case of the badi method is to "link" the response message
*    and the request message in a synchronouse communicaiton scenario in customer specific logic
  GET BADI lr_runtime_badi.
  IF lr_runtime_badi IS BOUND.
    lv_guid_32 = lv_msgguid_sender.
    CALL BADI lr_runtime_badi->transfer_response_guid
      EXPORTING
        iv_ns          = ns
        iv_ifname      = ifname
        iv_ifversion   = ifversion
        iv_request_guid  = lv_guid_32
        iv_response_guid = lv_new_guid.
  ENDIF.
  "Note 2151669  store outbound interface related global values
  lv_datum = gv_msgdate.
  lv_uzeit = gv_msgtime.
  ls_finf = gs_finf.

*   call main processing function for respones message
  CALL FUNCTION '/AIF/FILE_PROCESS_DATA'
    EXPORTING
      ns                = ls_inbound_finf-ns
      ifname            = ls_inbound_finf-ifname
      ifversion         = ls_inbound_finf-ifversion
      ximsgguid         = ximsgguid
      xi_flag           = 'X'
      testrun           = ls_inbound_finf-testrun
      pipelineid        = lv_pipelineid
    TABLES
      return_tab        = lt_message_return
    CHANGING
      data              = resp_sap_struct
```

```
      raw_struct           = <lrs_xi_param_input>
    EXCEPTIONS
      not_found            = 1
      customizing_incomplete = 2
      max_errors_reached     = 3
      cancel               = 4
      OTHERS               = 5.
  GET REFERENCE OF <lrs_xi_param_input> INTO resp_raw_struct.
  es_inbound_finf = ls_inbound_finf. " Note 2144871 return the inbound interface, so that the interface can be persisted wi
*   Note 2144871: once the reponse was processed add a log message to the request message
  MESSAGE w199(/aif/mes) WITH ls_inbound_finf-ns ls_inbound_finf-ifname ls_inbound_finf-ifversion INTO lv_dummy_message. " where-used-list
  lr_enabler->add_single_log_message( iv_msg_type   = 'I'
                                      iv_msg_id     = '/AIF/MES'
                                      iv_msg_number = '199'
                                      iv_msg_message_v1 = ls_inbound_finf-ns
                                      iv_msg_message_v2 = ls_inbound_finf-ifname
                                      iv_msg_message_v3 = ls_inbound_finf-ifversion
                                      iv_msg_message_v4 = lv_new_guid
                                      is_bal_context    = ls_bal_context
                                      iv_ignore_trace_level = 'X' ).
  "Note 2151669  restore outbound interface related global values
  gv_ns        = ls_finf-ns."  outbound interface namespace
  gv_ifname    = ls_finf-ifname." outbound interface
  gv_ifversion  = ls_finf-ifversion." outbound interface version
  gs_finf      = ls_finf. " outbound interface definition
*   gv_msgguid    = lv_msgguid_sender." outbound (request) message GUID
  gv_msgdate    = lv_datum.
  gv_msgtime    = lv_uzeit.
  "Note 2151669
 ENDIF.

 IF NOT lv_error IS INITIAL." OR ( NOT lv_sync IS INITIAL AND NOT sy-subrc IS INITIAL ).
  MESSAGE e110(/aif/mes) INTO lv_dummy_message. " where-used-list
  lr_enabler->add_single_log_message( iv_msg_type   = 'E'
```

```
                        iv_msg_id     = '/AIF/MES'
                        iv_msg_number = '110'
                        iv_msg_message    = lv_dummy_message
                        is_bal_context    = ls_bal_context ).
  TRY.
    lr_enabler->/aif/if_enabler_base~update(
      iv_message_status_flag  = 'E'
      iv_message_aif_location = lv_current_message_location
      iv_do_commit          = do_commit ).
   CATCH /aif/cx_enabler_base .
    MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
  ENDTRY.
  MESSAGE e530(sy) RAISING general_error.
 ENDIF.

 " this is the final call to the enabler update -> do not set status flag to derive it from messages
 " and make sure it has a final state
 IF lv_pipelineid EQ 'SENDER'
   OR /aif/cl_proxy_outbound=>is_local_logical_port( iv_lp_name = logical_port_name iv_class_name = ls_finf-proxyclassnamecl ) EQ 'X'
   OR ( NOT lv_sync IS INITIAL AND lv_pipelineid EQ 'WS_SENDER' ).
  TRY.
    lr_enabler->/aif/if_enabler_base~update(
      " no message status provided -> FINAL status will be derived from messages in app log
      iv_message_aif_location = lv_current_message_location
      iv_do_commit          = do_commit
      ).
   CATCH /aif/cx_enabler_base .
    MESSAGE e197(/aif/error_handling) RAISING status_update_failed.
  ENDTRY.
 ENDIF.
ENDFUNCTION.
```

Appendix - 3

Reverts the number range after failure case

```
FUNCTION z_fi_edoc_chile_upd_no_range.

*"----------------------------------------------------------------------

*"*"Local Interface:

*"  IMPORTING

*"     VALUE(IV_DTE_NUMBER) TYPE  EDOC_CL_NUMBER OPTIONAL

*"     VALUE(IV_SUBOBJ) TYPE  NRSOBJ OPTIONAL

*"     VALUE(IV_DTE_TYPE) TYPE  EDOC_CL_DTE OPTIONAL
```

```
*"  EXPORTING
*"     VALUE(EV_SUCCESS) TYPE  BOOLEAN
*"----------------------------------------------------------------------


  CONSTANTS: lc_name TYPE string VALUE 'EDOC_CL'.


  DATA:
    ls_inriv        TYPE inriv,
    lv_object       TYPE nrobj,
    lt_error_iv     TYPE STANDARD TABLE OF inriv,
    lt_interval     TYPE STANDARD TABLE OF inriv,
    lv_error_occured TYPE boole_d.


    lv_object = lc_name && iv_dte_type.


  SELECT * FROM nriv INTO TABLE @DATA(lt_nriv) WHERE object = @lv_object AND
                                       subobject = @iv_subobj .
  IF sy-subrc = 0.
    TRY.
       DATA(ls_nriv) = lt_nriv[ 1 ].
     CATCH cx_sy_itab_line_not_found.
       CLEAR: ls_nriv.
    ENDTRY.
  ENDIF.


*/Enqueue number range object
  CALL FUNCTION 'NUMBER_RANGE_ENQUEUE'
    EXPORTING
      object          = lv_object
    EXCEPTIONS
      foreign_lock     = 1
      object_not_found = 2
      system_failure   = 3
      OTHERS           = 4.
```

```
 IF sy-subrc IS NOT INITIAL.
*/  Leave operation
   ev_success = abap_false.
   RETURN.
 ENDIF.

*/Generate entry
 ls_inriv-toyear     = ls_nriv-toyear .
 ls_inriv-nrrangenr  = ls_nriv-nrrangenr .
 ls_inriv-subobject  = ls_nriv-subobject .
 ls_inriv-externind  = space.
 ls_inriv-procind    = 'U'.
 ls_inriv-fromnumber = ls_nriv-fromnumber.
 ls_inriv-tonumber   = ls_nriv-tonumber.
 ls_inriv-nrlevel    = iv_dte_number.

 INSERT ls_inriv INTO TABLE lt_interval.

*/Update interval
 CALL FUNCTION 'NUMBER_RANGE_INTERVAL_UPDATE'
   EXPORTING
     object          = lv_object
     subobject       = iv_subobj
   IMPORTING
     error_occured    = lv_error_occured
   TABLES
     error_iv        = lt_error_iv
     interval        = lt_interval
   EXCEPTIONS
     object_not_found = 01.

 IF     lv_error_occured IS INITIAL
    AND sy-subrc        IS INITIAL.
   CALL FUNCTION 'NUMBER_RANGE_UPDATE_CLOSE'
     EXPORTING
```

```
      object               = lv_object
    EXCEPTIONS
      no_changes_made      = 0
      object_not_initialized = 1.
  IF sy-subrc IS NOT INITIAL.
    ev_success = abap_false.
  ELSE.
    ev_success = abap_true.
  ENDIF.
 ELSE.
  ev_success = abap_false.
 ENDIF.
*/Dequeue number range object
 CALL FUNCTION 'NUMBER_RANGE_DEQUEUE'
  EXPORTING
    object = lv_object.
ENDFUNCTION.
```

POST- Exit method to update E-document

```
  CONSTANTS : lc_cl    TYPE land1 VALUE 'CL',
              lc_snda  TYPE edoc_status  VALUE 'SNDA',
              lc_apps  TYPE edoc_status  VALUE 'APPS',
              lc_status TYPE edoc_docstat VALUE 'XXXX         X                    '.

  IF cs_edocument-land = lc_cl.
    IF cs_edocument-proc_status = lc_snda.
      SELECT edoc_guid , file_name
        FROM zfid_edoc_pdf_cl  INTO TABLE @DATA(lt_pdf)
        WHERE edoc_guid = @cs_edocument-edoc_guid.
      IF sy-subrc = 0.
** Form data in above table confirms Approved by SII
        cs_edocument-proc_status = lc_apps.
        cs_edocument-status = lc_status.
      ENDIF.

    ENDIF.
```

```
  ENDIF.
```

Function Module for PDF response

```
FUNCTION z_fi_edoc_chile_pdf_entry.

*"----------------------------------------------------------------------

*"*"Local Interface:

*"  IMPORTING

*"     VALUE(LS_ENTRY) TYPE  ZFID_EDOC_PDF_CL OPTIONAL

*"----------------------------------------------------------------------


 IF ls_entry IS NOT INITIAL.

   MODIFY zfid_edoc_pdf_cl FROM ls_entry.

 ENDIF.


ENDFUNCTION.
```

Display PDF response

```
*  This method generates a PDF file based on the eDocument XML,

*  downloads the file in the SAP GUI temp local folder, opens the PDF on the front end

*  and once the file is closed deletes it from the temp folder


   DATA: lo_pdf          TYPE REF TO cl_edocument_pdf,

         lo_edocument_db   TYPE REF TO cl_edocument_db,

         lo_edocument      TYPE REF TO cl_edocument,      "2656570

         ls_edocfile       TYPE edocumentfile,

         ls_edocument      TYPE edocument,

         lv_file_type      TYPE edoc_file_type,          "2656570

         lv_file_raw       TYPE edoc_file,               "2656570

         lv_pdf_create     TYPE abap_bool,               "2822897

         lt_message        TYPE edoc_syst_msg_tab,       "2822897

         lv_pdf            TYPE xstring,

         lt_tab            TYPE tsfixml,

         lv_file_path      TYPE string,

         lv_rc             TYPE sysubrc,

         lv_file_name      TYPE string,

         lv_file_extension TYPE string,
```

```
      lv_file_separator TYPE c,
      lv_len           TYPE i.

  CLEAR ls_edocument.
  CLEAR lv_file_name.                          "2822897

*   Action only allowed for single eDocument
  READ TABLE it_edocument INTO ls_edocument INDEX 1.

  IF ls_edocument-land = 'CL'.
** Start Custom Logic
    DATA: fic TYPE xstring,
          tot TYPE xstring,
          str TYPE string,
          stt TYPE c LENGTH 2624.
    DATA: lt_solix TYPE TABLE OF solix,
          fil_len  TYPE i,
          it_solix TYPE TABLE OF solix.


    TYPES: BEGIN OF ly_line,
             tdline(128),
           END OF ly_line.

    DATA: lt_xstr TYPE STANDARD TABLE OF xstring,
          line    TYPE STANDARD TABLE OF  ly_line,
          ls_line TYPE ly_line,
          lt_line TYPE STANDARD TABLE OF  ly_line,
          it_line TYPE STANDARD TABLE OF  ly_line.

    SELECT * FROM zfid_edoc_pdf_cl
     INTO TABLE @DATA(lt_tmp_file)
     WHERE edoc_guid = @ls_edocument-edoc_guid.
    IF lt_tmp_file IS NOT INITIAL.
      SORT lt_tmp_file BY seq_no DESCENDING.
```

```
  READ TABLE lt_tmp_file ASSIGNING FIELD-SYMBOL(<ls_tmp>) INDEX 1.
  IF sy-subrc = 0.
    lv_file_raw  = <ls_tmp>-file_raw.
    lv_file_name = <ls_tmp>-file_name.              "2822897
  ENDIF.
 ENDIF.

 IF lv_file_raw IS NOT INITIAL.
  DATA(lv_initial_len) = 0.
  DATA(lv_final_len) = 128.

  DATA: lv_length TYPE i,
        e_string  TYPE string,
        lt_binary TYPE STANDARD TABLE OF x255.

  CALL FUNCTION 'SCMS_XSTRING_TO_BINARY'
    EXPORTING
      buffer       = lv_file_raw
    IMPORTING
      output_length = lv_length
    TABLES
      binary_tab    = lt_binary.

  CALL FUNCTION 'SCMS_BINARY_TO_STRING'
    EXPORTING
      input_length = lv_length
    IMPORTING
      text_buffer  = e_string
    TABLES
      binary_tab   = lt_binary
    EXCEPTIONS
      failed       = 1
      OTHERS       = 2.
  IF sy-subrc <> 0.
*    Implement suitable error handling here
```

```
      ENDIF.
      DATA(lv_len1) = strlen( e_string ).
      DO.
       IF ( lv_initial_len + lv_final_len ) > lv_len1.
        lv_final_len = lv_len1 - lv_initial_len.
       ENDIF.
       ls_line = e_string+lv_initial_len(lv_final_len).
       APPEND ls_line TO line.
       CLEAR: ls_line.
       IF ( lv_final_len + lv_initial_len ) = lv_len1.
        EXIT.
       ENDIF.
       lv_initial_len = lv_initial_len + 128.
      ENDDO.
*
*     DESCRIBE TABLE line LINES DATA(llll).
*
      LOOP AT line ASSIGNING FIELD-SYMBOL(<ls>).
*  REPLACE ALL OCCURRENCES OF '#' IN <ls> WITH ''.
       str = <ls>.
*  if llll = sy-tabix.
*   break mbhardwaj.
*   endif.
*  str = stt.
*    CALL FUNCTION 'SSFC_BASE64_DECODE'
*     EXPORTING
*      b64data = str
*     IMPORTING
*      bindata = fic.
*      CALL METHOD cl_http_utility=>if_http_utility~decode_base64
*       EXPORTING
*         encoded = str
*       RECEIVING
*         decoded = str.
       CALL METHOD cl_http_utility=>if_http_utility~decode_x_base64
```

```
          EXPORTING
            encoded = str
          RECEIVING
            decoded = fic.

        tot = tot && fic.

        CALL FUNCTION 'CONVERT_STRING_TO_TABLE'
          EXPORTING
            i_string        = str
            i_tabline_length = 96
*           I_UNICODE       =
          TABLES
            et_table        = lt_line.
        APPEND LINES OF lt_line TO it_line.
        CLEAR: lt_line.
*fic = str.
        APPEND fic TO lt_xstr.
        CALL FUNCTION 'SCMS_XSTRING_TO_BINARY'
          EXPORTING
            buffer     = fic
*           APPEND_TO_TABLE      = ' '
* IMPORTING
*           OUTPUT_LENGTH        =
          TABLES
            binary_tab = it_solix.

        DATA: lv_sss TYPE string.
        lv_sss = CONV #( fic ).

*  CALL METHOD cl_bcs_convert=>xstring_to_solix
*    EXPORTING
*      iv_xstring = fic
*    RECEIVING
*      et_solix   = it_solix.
```

```
        APPEND LINES OF it_solix TO lt_solix.
        CLEAR: it_solix.
      ENDLOOP.

*  METHOD convert_xstring_to_string.

      lv_pdf = tot.

      IF lv_pdf IS NOT INITIAL.
*  Get local SAP GUI temporary folder
        CALL METHOD cl_gui_frontend_services=>get_temp_directory
          CHANGING
            temp_dir           = lv_file_path
          EXCEPTIONS
            cntl_error         = 1
            error_no_gui       = 2
            not_supported_by_gui = 3
            OTHERS             = 4.
        IF sy-subrc IS NOT INITIAL.
          MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
             WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 INTO cl_edocument=>gv_error_txt.
          cl_edocument=>raise_edoc_exception( ).
        ENDIF.

        CALL METHOD cl_gui_cfw=>flush
          EXCEPTIONS
            cntl_system_error = 1
            cntl_error       = 2
            OTHERS           = 3.
        IF sy-subrc IS NOT INITIAL.
          MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
             WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 INTO cl_edocument=>gv_error_txt.
          cl_edocument=>raise_edoc_exception( ).
        ENDIF.
```

```
* get file separator
        CALL METHOD cl_gui_frontend_services=>get_file_separator
         CHANGING
           file_separator = lv_file_separator.
        IF lv_file_separator IS INITIAL.
         lv_file_separator = '/'.
        ENDIF.

* Set PDF file name the same as the XML file name
        IF lv_file_name IS INITIAL.                    "2822897
         CREATE OBJECT lo_edocument_db.
         ls_edocfile = lo_edocument_db->if_edocument_db~select_edocumentfile( ls_edocument-file_guid ).
         lv_file_name = ls_edocfile-file_name.          "2822897
        ENDIF.                                         "2822897

        SPLIT lv_file_name AT '.' INTO lv_file_name lv_file_extension. "2822897
        CONCATENATE lv_file_path lv_file_separator lv_file_name '.PDF' INTO lv_file_path.

        CALL FUNCTION 'SCMS_XSTRING_TO_BINARY'
         EXPORTING
           buffer       = lv_pdf
         IMPORTING
           output_length = lv_len
         TABLES
           binary_tab    = lt_tab.

* Download a temporary PDF to the SAP GUI temp folder
        CALL METHOD cl_gui_frontend_services=>gui_download
         EXPORTING
           bin_filesize          = lv_len
           filename              = lv_file_path
           filetype              = 'BIN'
         CHANGING
           data_tab              = lt_tab
```

```
      EXCEPTIONS
        file_write_error       = 1
        no_batch               = 2
        gui_refuse_filetransfer = 3
        invalid_type           = 4
        no_authority           = 5
        unknown_error          = 6
        header_not_allowed     = 7
        separator_not_allowed  = 8
        filesize_not_allowed   = 9
        header_too_long        = 10
        dp_error_create        = 11
        dp_error_send          = 12
        dp_error_write         = 13
        unknown_dp_error       = 14
        access_denied          = 15
        dp_out_of_memory       = 16
        disk_full              = 17
        dp_timeout             = 18
        file_not_found         = 19
        dataprovider_exception  = 20
        control_flush_error    = 21
        not_supported_by_gui   = 22
        error_no_gui           = 23
        OTHERS                 = 24.
      IF sy-subrc IS NOT INITIAL.
       MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
          WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 INTO cl_edocument=>gv_error_txt.
       cl_edocument=>raise_edoc_exception( ).
      ENDIF.
*  Open the PDF
      CALL METHOD cl_gui_frontend_services=>execute
       EXPORTING
        document               = lv_file_path
        synchronous            = 'X'
```

```
      EXCEPTIONS
       cntl_error           = 1
       error_no_gui         = 2
       bad_parameter        = 3
       file_not_found       = 4
       path_not_found       = 5
       file_extension_unknown = 6
       error_execute_failed   = 7
       synchronous_failed     = 8
       not_supported_by_gui   = 9
       OTHERS               = 10.
      IF sy-subrc IS NOT INITIAL.
       MESSAGE ID sy-msgid TYPE 'I' NUMBER sy-msgno
          WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 INTO cl_edocument=>gv_error_txt.
       cl_edocument=>raise_edoc_exception( ).
      ENDIF.
*  Delete the temporary PDF
      CALL METHOD cl_gui_frontend_services=>file_delete
       EXPORTING
        filename           = lv_file_path
       CHANGING
        rc                 = lv_rc
       EXCEPTIONS
        file_delete_failed   = 1
        cntl_error         = 2
        error_no_gui       = 3
        file_not_found     = 4
        access_denied      = 5
        unknown_error      = 6
        not_supported_by_gui = 7
        wrong_parameter    = 8
        OTHERS             = 9.
      IF sy-subrc IS NOT INITIAL.
*     No error if the temporary file was not deleted
      ENDIF.
```

```
      ENDIF.


    ELSE.
*   No file found for eDoc GUID
      MESSAGE e185(edocument) INTO cl_edocument=>gv_error_txt.
      cl_edocument=>raise_edoc_exception( ).
    ENDIF.


** End Custom Logic
  ELSE.
    IF ls_edocument-edoc_guid IS NOT INITIAL.
     IF is_edocumentfile IS INITIAL.                  "2822897
       cl_edocument=>retrieve_by_edoc_guid(           "2656570
       EXPORTING iv_edoc_guid = ls_edocument-edoc_guid   "2656570
       RECEIVING ro_edocument = lo_edocument ).         "2656570
       IF lo_edocument IS BOUND.                        "2656570
        lv_pdf_create = abap_true.                    "2822897
        lo_edocument->get_file_for_pdf(               "2656570
          IMPORTING ev_file_type = lv_file_type        "2656570
                    ev_file_raw  = lv_file_raw         "2656570
                    ev_file_name = lv_file_name       "2822897
                    ev_pdf_file  = lv_pdf            "2822897
          CHANGING  cv_pdf_create = lv_pdf_create  ). "2822897
       ENDIF.                                        "2656570

       IF lv_pdf IS INITIAL.                                "start 2822897
        IF lv_pdf_create EQ abap_true.
          CREATE OBJECT lo_pdf EXPORTING is_edocument = ls_edocument.
          lv_pdf = lo_pdf->create_pdf( iv_file_type = lv_file_type "2656570
                                       iv_file_raw  = lv_file_raw ). "2656570
        ELSE.
          MESSAGE e229(edocument) INTO cl_edocument=>gv_error_txt.
          cl_edocument=>raise_edoc_exception( ).
        ENDIF.
       ENDIF.                                             "end 2822897
```

```
      ELSE.                                  "2822897
       lv_pdf = is_edocumentfile-file_raw.             "2822897
       lv_file_type = is_edocumentfile-file_type.      "2822897
       lv_file_name = is_edocumentfile-file_name.      "2822897
      ENDIF.                                 "2822897



      IF lv_pdf IS NOT INITIAL.
*  Get local SAP GUI temporary folder
       CALL METHOD cl_gui_frontend_services=>get_temp_directory
        CHANGING
         temp_dir           = lv_file_path
        EXCEPTIONS
         cntl_error         = 1
         error_no_gui       = 2
         not_supported_by_gui = 3
         OTHERS             = 4.
       IF sy-subrc IS NOT INITIAL.
        MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
           WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 INTO cl_edocument=>gv_error_txt.
        cl_edocument=>raise_edoc_exception( ).
       ENDIF.

       CALL METHOD cl_gui_cfw=>flush
        EXCEPTIONS
         cntl_system_error = 1
         cntl_error       = 2
         OTHERS           = 3.
       IF sy-subrc IS NOT INITIAL.
        MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
           WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 INTO cl_edocument=>gv_error_txt.
        cl_edocument=>raise_edoc_exception( ).
       ENDIF.
```

```
* get file separator
        CALL METHOD cl_gui_frontend_services=>get_file_separator
         CHANGING
           file_separator = lv_file_separator.
        IF lv_file_separator IS INITIAL.
         lv_file_separator = '/'.
        ENDIF.

* Set PDF file name the same as the XML file name
        IF lv_file_name IS INITIAL.                    "2822897
         CREATE OBJECT lo_edocument_db.
         ls_edocfile = lo_edocument_db->if_edocument_db~select_edocumentfile( ls_edocument-file_guid ).
         lv_file_name = ls_edocfile-file_name.          "2822897
        ENDIF.                                         "2822897

        SPLIT lv_file_name AT '.' INTO lv_file_name lv_file_extension. "2822897
        CONCATENATE lv_file_path lv_file_separator lv_file_name '.PDF' INTO lv_file_path.

        CALL FUNCTION 'SCMS_XSTRING_TO_BINARY'
         EXPORTING
           buffer       = lv_pdf
         IMPORTING
           output_length = lv_len
         TABLES
           binary_tab    = lt_tab.

* Download a temporary PDF to the SAP GUI temp folder
        CALL METHOD cl_gui_frontend_services=>gui_download
         EXPORTING
           bin_filesize          = lv_len
           filename              = lv_file_path
           filetype              = 'BIN'
         CHANGING
           data_tab              = lt_tab
         EXCEPTIONS
```

```
      file_write_error       = 1
      no_batch               = 2
      gui_refuse_filetransfer = 3
      invalid_type           = 4
      no_authority           = 5
      unknown_error          = 6
      header_not_allowed     = 7
      separator_not_allowed  = 8
      filesize_not_allowed   = 9
      header_too_long        = 10
      dp_error_create        = 11
      dp_error_send          = 12
      dp_error_write         = 13
      unknown_dp_error       = 14
      access_denied          = 15
      dp_out_of_memory       = 16
      disk_full              = 17
      dp_timeout             = 18
      file_not_found         = 19
      dataprovider_exception  = 20
      control_flush_error    = 21
      not_supported_by_gui   = 22
      error_no_gui           = 23
      OTHERS                 = 24.
    IF sy-subrc IS NOT INITIAL.
     MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
        WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 INTO cl_edocument=>gv_error_txt.
     cl_edocument=>raise_edoc_exception( ).
    ENDIF.
* Open the PDF
    CALL METHOD cl_gui_frontend_services=>execute
     EXPORTING
      document             = lv_file_path
      synchronous          = 'X'
     EXCEPTIONS
```

```
          cntl_error           = 1
          error_no_gui         = 2
          bad_parameter        = 3
          file_not_found       = 4
          path_not_found       = 5
          file_extension_unknown = 6
          error_execute_failed   = 7
          synchronous_failed     = 8
          not_supported_by_gui   = 9
          OTHERS               = 10.
        IF sy-subrc IS NOT INITIAL.
         MESSAGE ID sy-msgid TYPE 'I' NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4 INTO cl_edocument=>gv_error_txt.
         cl_edocument=>raise_edoc_exception( ).
        ENDIF.
*  Delete the temporary PDF
        CALL METHOD cl_gui_frontend_services=>file_delete
         EXPORTING
          filename           = lv_file_path
         CHANGING
          rc                 = lv_rc
         EXCEPTIONS
          file_delete_failed   = 1
          cntl_error           = 2
          error_no_gui         = 3
          file_not_found       = 4
          access_denied        = 5
          unknown_error        = 6
          not_supported_by_gui = 7
          wrong_parameter      = 8
          OTHERS               = 9.
        IF sy-subrc IS NOT INITIAL.
*       No error if the temporary file was not deleted
         ENDIF.
       ENDIF.
```

```
  ENDIF.
ENDIF.
```