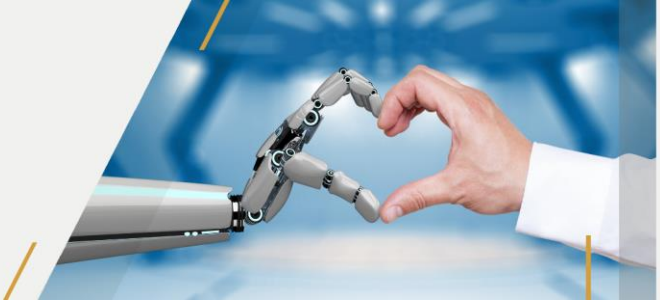

PERFORMER SUITE

Customer Function Modules

SCENARIO VERSION	1.0
SETTINGS VARIANT	IT STANDARD DOCUMENTATION
COMMENT VARIANT	---
RESPONSIBLE PERSON	ALEXANDER DÜRRSTEIN
CREATION DATE	11.08.2020

Table of contents

1 Docu Performer General Settings

Performer Suite Settings	
Settings Variant	IT standard documentation
Comment Variant	
Word Template	Performer_Suite_Scenario_Template_EN.dotx

2 Function Module Z_RFC_ADSO_CREATE_XML - Anlegen eines ADSOs

2.1 General Information

System	BI2
Function Module	Z_RFC_ADSO_CREATE_XML, Anlegen eines ADSOs
Function Pool	Z_DP
Remote	Yes
Last changed by	NMEYER
Last change (timestamp)	11/06/2018
Timestamp of documentation	11/08/2020 16:29:22

2.2 Import Parameter

Parameter		**Associated Type**	**Default**	**Opt.**	**Pass**	**Short text**
I_ADSONM	TYPE	RSOADSONM			X	Datastore Object: Name
I_TEXT	TYPE	RSOADSODESCR		X	X	Datastore Objects: Description
I_INFOAREA	TYPE	RSINFOAREA		X	X	InfoArea
IV_ADSOFLAGS	TYPE	STRING			X	Datastore Object Flags
IV_OBJECT	TYPE	STRING			X	
IV_DIMENSION	TYPE	STRING			X	
IV_KEY	TYPE	STRING			X	
IV_HASH	TYPE	STRING			X	
IV_INDEX	TYPE	STRING			X	
IV_PARTITION	TYPE	STRING			X	
IV_CHA_CONST	TYPE	STRING			X	
IV_VALIDITY	TYPE	STRING			X	

2.3 Exceptions

Exception	**Short text**
NOT_ACTIVE	konnte nicht in aktiver Version angelegt werden

NOT_CREATED	konnte nicht angelegt werden
RELEASE_1_2	

2.4 Source code Function module

```
FUNCTION Z_RFC_ADSO_CREATE_XML.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(I_ADSONM) TYPE  RSOADSONM
*"     VALUE(I_TEXT) TYPE  RSOADSODESCR OPTIONAL
*"     VALUE(I_INFOAREA) TYPE  RSINFOAREA OPTIONAL
*"     VALUE(IV_ADSOFLAGS) TYPE  STRING
*"     VALUE(IV_OBJECT) TYPE  STRING
*"     VALUE(IV_DIMENSION) TYPE  STRING
*"     VALUE(IV_KEY) TYPE  STRING
*"     VALUE(IV_HASH) TYPE  STRING
*"     VALUE(IV_INDEX) TYPE  STRING
*"     VALUE(IV_PARTITION) TYPE  STRING
*"     VALUE(IV_CHA_CONST) TYPE  STRING
*"     VALUE(IV_VALIDITY) TYPE  STRING
*"  EXCEPTIONS
*"      NOT_ACTIVE
*"      NOT_CREATED
*"      RELEASE_1_2
*"----------------------------------------------------------------------
   DATA: lt_resbind    TYPE abap_trans_resbind_tab.
   DATA: ls_resbind    TYPE abap_trans_resbind.
   DATA: lt_parameters TYPE abap_parmbind_tab.
   DATA: ls_parameter  TYPE abap_parmbind.
   DATA: lo_ref        TYPE REF TO data.

   FIELD-SYMBOLS: <any> TYPE any.

   " get method parameter types
   SELECT
   sconame,
   type
   FROM seosubcodf
   INTO TABLE @DATA(lt_method_param_types)
   WHERE clsname = 'CL_RSO_ADSO_API'
     AND cmpname = 'CREATE'
     AND type <> ''
   ORDER BY editorder.

   LOOP AT lt_method_param_types ASSIGNING FIELD-SYMBOL(<ls_method_param_type>).
     FREE: ls_parameter, ls_resbind, lt_resbind.

     TRY. " create data reference for parameter
         CREATE DATA lo_ref TYPE (<ls_method_param_type>-type).
       CATCH cx_sy_create_data_error.
         " probably type of class
         DATA(lv_type) = |CL_RSO_ADSO_API=>{ <ls_method_param_type>-type }|.
         TRY.
             CREATE DATA lo_ref TYPE (lv_type).
           CATCH cx_sy_create_data_error..
             " no plan c
             EXIT.
         ENDTRY.
     ENDTRY.

     ls_resbind = VALUE abap_trans_resbind(
         name  = <ls_method_param_type>-sconame
         value = lo_ref  ).
     INSERT ls_resbind INTO TABLE lt_resbind.

     " determine kind of parameter for dynamic method call
     CASE <ls_method_param_type>-sconame+0(1).
       WHEN 'E'.
         DATA(lv_kind) = cl_abap_objectdescr=>importing.
       WHEN 'I'.
         lv_kind =  cl_abap_objectdescr=>exporting.
       WHEN OTHERS.
         EXIT.
     ENDCASE.

     CASE <ls_method_param_type>-sconame.
```

```
    WHEN 'I_ADSONM'.
      GET REFERENCE OF i_adsonm INTO DATA(lo_adsonmref).
      ls_parameter = VALUE abap_parmbind(
          name  = <ls_method_param_type>-sconame
          kind  = lv_kind
          value = lo_adsonmref ).
      INSERT ls_parameter INTO TABLE lt_parameters.
    WHEN 'I_TEXT'.
      GET REFERENCE OF i_text INTO DATA(lo_text).
      ls_parameter = VALUE abap_parmbind(
          name  = <ls_method_param_type>-sconame
          kind  = lv_kind
          value = lo_text ).
      INSERT ls_parameter INTO TABLE lt_parameters.
      CONTINUE.
    WHEN 'I_INFOAREA'.
      GET REFERENCE OF i_infoarea INTO DATA(lo_infoarea).
      ls_parameter = VALUE abap_parmbind(
          name  = <ls_method_param_type>-sconame
          kind  = lv_kind
          value = lo_infoarea ).
      INSERT ls_parameter INTO TABLE lt_parameters.
      CONTINUE.
    WHEN 'I_S_ADSOFLAGS'.
      CALL TRANSFORMATION (`ID`) SOURCE XML iv_adsoflags RESULT (lt_resbind).
    WHEN 'I_T_OBJECT'.
      CALL TRANSFORMATION (`ID`) SOURCE XML iv_object RESULT (lt_resbind).
    WHEN 'I_T_DIMENSION'.
      CALL TRANSFORMATION (`ID`) SOURCE XML iv_dimension RESULT (lt_resbind).
    WHEN 'I_T_KEY'.
      CALL TRANSFORMATION (`ID`) SOURCE XML iv_key RESULT (lt_resbind).
    WHEN 'I_T_HASH'.
      CALL TRANSFORMATION (`ID`) SOURCE XML iv_hash RESULT (lt_resbind).
    WHEN 'I_T_INDEX'.
      CALL TRANSFORMATION (`ID`) SOURCE XML iv_index RESULT (lt_resbind).
    WHEN 'I_T_PARTITION'.
      CALL TRANSFORMATION (`ID`) SOURCE XML iv_partition RESULT (lt_resbind).
    WHEN 'I_T_CHA_CONST'.
      CALL TRANSFORMATION (`ID`) SOURCE XML iv_cha_const RESULT (lt_resbind).
    WHEN 'I_T_VALIDITY'.
      CALL TRANSFORMATION (`ID`) SOURCE XML iv_validity RESULT (lt_resbind).
  ENDCASE.

  " build parameter table for method call
  ls_parameter = VALUE abap_parmbind(
      name  = <ls_method_param_type>-sconame
      kind  = lv_kind
      value = lt_resbind[ 1 ]-value ).
  INSERT ls_parameter INTO TABLE lt_parameters.

ENDLOOP.

TRY.
    CALL METHOD ('CL_RSO_ADSO_API')=>create PARAMETER-TABLE lt_parameters.
  CATCH cx_root. " catch all errors and exit
ENDTRY.

"check if object is created, active and dequeue it
DATA: lv_exists TYPE i.
SELECT COUNT( * )
 FROM rsoadso
   INTO lv_exists
     WHERE adsonm  = i_adsonm
     AND ( objvers   = 'A'
     OR objvers    = 'M' ).

IF NOT lv_exists >= 1.
  RAISE not_created.
ENDIF.

DATA: lo_adso TYPE REF TO cl_rso_adso.

CALL METHOD cl_rso_adso=>factory
  EXPORTING
    i_adsonm = i_adsonm
  RECEIVING
    r_r_adso = lo_adso.
```

```
   "dequeue
   CALL METHOD lo_adso->if_rso_tlogo_maintain~dequeue.

   "is active
   DATA: is_active TYPE rs_bool.

   CALL METHOD lo_adso->if_rso_tlogo_maintain~is_active
     RECEIVING
       r_is_active = is_active.

   IF is_active = rs_c_false.
     RAISE not_active.
   ENDIF.
 ENDFUNCTION.
```

3 Function Module Z_RFC_ADSO_GETDTL_XML - Z_RFC_ADSO_GETDTL_XML

3.1 General Information

System	BI2
Function Module	Z_RFC_ADSO_GETDTL_XML, Z_RFC_ADSO_GETDTL_XML
Function Pool	Z_DP
Remote	Yes
Last changed by	TSCHMIDT
Last change (timestamp)	01/08/2017
Timestamp of documentation	11/08/2020 16:29:26

3.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
I_ADSONM	TYPE	RSOADSONM			X	

3.3 Export Parameter

Parameter		Associated Type	Pass	Short text
E_TEXT	TYPE	RSOADSODESCR	X	
E_INFOAREA	TYPE	RSINFOAREA	X	
EV_ADSOFLAGS_XML	TYPE	STRING	X	
EV_OBJECT_XML	TYPE	STRING	X	
EV_DIMENSION_XML	TYPE	STRING	X	
EV_KEY_XML	TYPE	STRING	X	
EV_HASH_XML	TYPE	STRING	X	
EV_INDEX_XML	TYPE	STRING	X	
EV_PARTITION_XML	TYPE	STRING	X	
EV_CHA_CONST_XML	TYPE	STRING	X	
EV_VALIDITY_XML	TYPE	STRING	X	

3.4 Exceptions

Exception	Short text
RELEASE_1_0	

3.5 Source code Function module

```
FUNCTION Z_RFC_ADSO_GETDTL_XML.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(I_ADSONM) TYPE  RSOADSONM
*"  EXPORTING
```

```
*"     VALUE(E_TEXT) TYPE  RSOADSODESCR
*"     VALUE(E_INFOAREA) TYPE  RSINFOAREA
*"     VALUE(EV_ADSOFLAGS_XML) TYPE  STRING
*"     VALUE(EV_OBJECT_XML) TYPE  STRING
*"     VALUE(EV_DIMENSION_XML) TYPE  STRING
*"     VALUE(EV_KEY_XML) TYPE  STRING
*"     VALUE(EV_HASH_XML) TYPE  STRING
*"     VALUE(EV_INDEX_XML) TYPE  STRING
*"     VALUE(EV_PARTITION_XML) TYPE  STRING
*"     VALUE(EV_CHA_CONST_XML) TYPE  STRING
*"     VALUE(EV_VALIDITY_XML) TYPE  STRING
*"  EXCEPTIONS
*"      RELEASE_1_0
*"----------------------------------------------------------------------
    DATA: lt_parameters TYPE abap_parmbind_tab.
    DATA: ls_parameter  TYPE abap_parmbind.
    DATA: lo_ref        TYPE REF TO data.

    FIELD-SYMBOLS: <any> TYPE any.

    " get method parameter types
    SELECT
    sconame,
    type
    FROM seosubcodf
    INTO TABLE @DATA(lt_method_param_types)
    WHERE clsname = 'CL_RSO_ADSO_API'
      AND cmpname = 'READ'
      AND type <> ''
    ORDER BY editorder.

    LOOP AT lt_method_param_types ASSIGNING FIELD-SYMBOL(<ls_method_param_type>).
      TRY .
          CREATE DATA lo_ref TYPE (<ls_method_param_type>-type).
        CATCH cx_sy_create_data_error.
          " probably type of class
          DATA(lv_type) = |CL_RSO_ADSO_API=>{ <ls_method_param_type>-type }|.
          TRY.
              CREATE DATA lo_ref TYPE (lv_type).
            CATCH cx_sy_create_data_error..
              " no plan c
              EXIT.
          ENDTRY.
      ENDTRY.

      CASE <ls_method_param_type>-sconame+0(1).
        WHEN 'E'.
          DATA(lv_kind) = cl_abap_objectdescr=>importing.
        WHEN 'I'.
          lv_kind =  cl_abap_objectdescr=>exporting.
        WHEN OTHERS.
          EXIT.
      ENDCASE.

      ls_parameter = VALUE abap_parmbind(
          name  = <ls_method_param_type>-sconame
          kind  = lv_kind
          value = lo_ref  ).
      INSERT ls_parameter INTO TABLE lt_parameters.
      FREE: ls_parameter.

    ENDLOOP.

    READ TABLE lt_parameters ASSIGNING FIELD-SYMBOL(<ls_parameter>) WITH KEY name = 'I_ADSONM'.
    IF sy-subrc = 0.
      ASSIGN <ls_parameter>-value->* TO <any>.
      <any> = i_adsonm.
    ENDIF.

    TRY.
        CALL METHOD ('CL_RSO_ADSO_API')=>read PARAMETER-TABLE lt_parameters.
      CATCH cx_root. " catch all errors and exit -> no exceptions defined
        EXIT.
    ENDTRY.

    " Transform to XML for Export
    LOOP AT lt_parameters ASSIGNING <ls_parameter>.
```

```
    ASSIGN <ls_parameter>-value->*  TO <any>.
    " For easier consumption in the create Module switch the naming in XML to I_
    " for the importing Parameters of the create method
    IF <ls_parameter>-name+0(1) = 'E'.
      DATA(lv_name) = |I{ <ls_parameter>-name+1(29) }|.
    ELSE.
      lv_name = <ls_parameter>-name.
    ENDIF.

    DATA(lt_srcbind) = VALUE abap_trans_srcbind_tab( (
    name  = lv_name
    value = <ls_parameter>-value ) ).

    CASE <ls_parameter>-name.
      WHEN 'E_TEXT'.
        e_text = <any>.
      WHEN 'E_INFOAREA'.
        e_infoarea = <any>..
      WHEN 'E_S_ADSOFLAGS'.
        CALL TRANSFORMATION (`ID`) SOURCE (lt_srcbind) RESULT XML ev_adsoflags_xml.
      WHEN 'E_T_OBJECT'.
        CALL TRANSFORMATION (`ID`) SOURCE (lt_srcbind) RESULT XML ev_object_xml.
      WHEN 'E_T_DIMENSION'.
        CALL TRANSFORMATION (`ID`) SOURCE (lt_srcbind) RESULT XML ev_dimension_xml.
      WHEN 'E_T_KEY'.
        CALL TRANSFORMATION (`ID`) SOURCE (lt_srcbind) RESULT XML ev_key_xml.
      WHEN 'E_T_HASH'.
        CALL TRANSFORMATION (`ID`) SOURCE (lt_srcbind) RESULT XML ev_hash_xml.
      WHEN 'E_T_INDEX'.
        CALL TRANSFORMATION (`ID`) SOURCE (lt_srcbind) RESULT XML ev_index_xml.
      WHEN 'E_T_PARTITION'.
        CALL TRANSFORMATION (`ID`) SOURCE (lt_srcbind) RESULT XML ev_partition_xml.
      WHEN 'E_T_CHA_CONST'.
        CALL TRANSFORMATION (`ID`) SOURCE (lt_srcbind) RESULT XML ev_cha_const_xml.
      WHEN 'E_T_VALIDITY'.
        CALL TRANSFORMATION (`ID`) SOURCE (lt_srcbind) RESULT XML ev_validity_xml.
    ENDCASE.
  ENDLOOP.

ENDFUNCTION.
```

4 Function Module Z_RFC_AUTH_CHECK - Check Authority RFC

4.1 General Information

System	BI2
Function Module	Z_RFC_AUTH_CHECK, Check Authority RFC
Function Pool	Z_DP
Remote	Yes
Last changed by	TSCHMIDT
Last change (timestamp)	24/08/2015
Timestamp of documentation	11/08/2020 16:29:29

4.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
I_COMPID	TYPE	XUVAL			X	Name (ID) of a reporting component
I_PROVIDER	TYPE	XUVAL			X	InfoProvider
I_INFOAREA	TYPE	XUVAL		X	X	InfoArea
I_ACTIVITY	TYPE	XUVAL			X	Activity
I_OWNER	TYPE	XUVAL			X	Authorization Value

4.3 Tables

Parameter		Associated Type	Opt.	Short text
T_USER	LIKE	BAPITGB		Structure to transfer texts for BAPIs that read docu.

4.4 Exceptions

Exception	Short text
RELEASE_1_0	

4.5 Source code Function module

```
FUNCTION Z_RFC_AUTH_CHECK.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_COMPID) TYPE  XUVAL
*"     VALUE(I_PROVIDER) TYPE  XUVAL
*"     VALUE(I_INFOAREA) TYPE  XUVAL OPTIONAL
*"     VALUE(I_ACTIVITY) TYPE  XUVAL
*"     VALUE(I_OWNER) TYPE  XUVAL
*"  TABLES
*"      T_USER STRUCTURE  BAPITGB
*"  EXCEPTIONS
*"      RELEASE_1_0
*"----------------------------------------------------------------------

  TYPES: BEGIN OF ty_user,
               bname TYPE xubname,
               name_text TYPE ad_namtext,
         END OF ty_user.

  FIELD-SYMBOLS <fs_user> TYPE ty_user.

  DATA: lt_user TYPE TABLE OF ty_user,
        chk_comp TYPE c LENGTH 1,
        chk_comp1 TYPE c LENGTH 1.

  REFRESH t_user.

  SELECT  bname name_text FROM user_addrp INTO TABLE lt_user.
  SORT lt_user.

  LOOP AT lt_user ASSIGNING <fs_user>.

    CLEAR: chk_comp, chk_comp1.

    CALL FUNCTION 'AUTHORITY_CHECK'
      EXPORTING
        user                = <fs_user>-bname
        object              = 'S_RS_COMP'
        field1              = 'RSZCOMPID'
        value1              = i_compid
        field2              = 'RSZCOMPTP'
        value2              = 'REP'
        field3              = 'RSINFOCUBE'
        value3              = i_provider
        field4              = 'ACTVT'
        value4              = i_activity
        field5              = 'RSINFOAREA'
        value5              = i_infoarea
      EXCEPTIONS
        user_dont_exist     = 1
        user_is_authorized  = 2
        user_not_authorized = 3
        user_is_locked      = 4
        OTHERS              = 5.
    IF sy-subrc = 2.
      chk_comp = 'X'.
    ENDIF.

    CALL FUNCTION 'AUTHORITY_CHECK'
      EXPORTING
        user                = <fs_user>-bname
        object              = 'S_RS_COMP1'
        field1              = 'RSZCOMPID'
        value1              = i_compid
        field2              = 'RSZCOMPTP'
        value2              = 'REP'
        field3              = 'RSZOWNER'
        value3              = i_owner
        field4              = 'ACTVT'
        value4              = i_activity
```

```
        field5              = ''
        value5              = ''
      EXCEPTIONS
        user_dont_exist     = 1
        user_is_authorized  = 2
        user_not_authorized = 3
        user_is_locked      = 4
        OTHERS              = 5.

    IF sy-subrc = 2.
      chk_comp1 = 'X'.
    ENDIF.

    IF chk_comp = 'X' AND chk_comp1 = 'X'.
      APPEND <fs_user> TO t_user.
    ENDIF.

  ENDLOOP.


ENDFUNCTION.
```

5 Function Module Z_RFC_CHECK_ACT_TR - Test RFC

5.1 General Information

System	BI2
Function Module	Z_RFC_CHECK_ACT_TR, Test RFC
Function Pool	Z_DP
Remote	Yes
Last changed by	ADUERRSTEIN
Last change (timestamp)	11/02/2019
Timestamp of documentation	11/08/2020 16:29:32

5.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
IV_MODE	TYPE	CHAR1	'1'		X	
IV_XML_OBJECTS	TYPE	STRING			X	
IV_TRANSPORT	TYPE	TRKORR		X	X	

5.3 Export Parameter

Parameter		Associated Type	Pass	Short text
EV_XML_STATUS	TYPE	STRING	X	

5.4 Exceptions

Exception	Short text
RELEASE_1_0	

5.5 Source code Function module

```
FUNCTION Z_RFC_CHECK_ACT_TR.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(IV_MODE) TYPE  CHAR1 DEFAULT '1'
*"     VALUE(IV_XML_OBJECTS) TYPE  STRING
*"     VALUE(IV_TRANSPORT) TYPE  TRKORR OPTIONAL
*"  EXPORTING
*"     VALUE(EV_XML_STATUS) TYPE  STRING
*"  EXCEPTIONS
*"      RELEASE_1_0
*"----------------------------------------------------------------------
  TYPES:
    BEGIN OF ty_message,
```

```
      message_type TYPE sy-msgty,
      message      TYPE string,
    END OF ty_message.
  TYPES: tt_messages TYPE STANDARD TABLE OF ty_message WITH DEFAULT KEY.

  TYPES:
    BEGIN OF ty_status,
      tlogo                TYPE rstlogo,
      objnm                TYPE sobj_name,
      actflg               TYPE rsawbn_obj_activflg,
      objstat              TYPE rsobjstat,
      saved                TYPE rs_bool,
      activflg             TYPE rsawbn_obj_activflg,
      locked               TYPE rs_bool,
      tr_lock              TYPE rs_bool,
      no_handler           TYPE rs_bool,
      written_to_tr        TYPE rs_bool,
      activation_successful TYPE rs_bool,
      messages             TYPE tt_messages,
    END OF ty_status.

  DATA: lt_objects        TYPE rso_t_tlogo.
  DATA: lt_status         TYPE TABLE OF ty_status.
  DATA: lv_objvers        TYPE rs_objvers.
  DATA: lo_tlogo          TYPE REF TO if_rso_tlogo.
  DATA: lv_objstat        TYPE rsobjstat.
  DATA: lo_msg            TYPE REF TO cl_rso_msg.
  DATA: lt_object_details TYPE rso_t_cto_tlogo_prop.
  DATA: lv_request        TYPE trkorr.
  DATA: lv_error          TYPE rs_bool.
  DATA: lt_tr_msg         TYPE bapirettab.
  DATA: lo_awbn_factory   TYPE REF TO cl_rsawbn_obj_factory.
  DATA: lo_awbn_object    TYPE REF TO cl_rsawbn_obj.
  DATA: lt_enq            TYPE TABLE OF seqg3.
  DATA: lv_garg           TYPE eqegraarg.
  DATA: lv_logsys         TYPE rsrlogsys.
  DATA: ls_tlogoprop      TYPE rstlogoprop.
  DATA: lt_msg            TYPE rs_t_msg.
  DATA: lv_repaired       TYPE rs_bool.
  DATA: lv_subrc          TYPE sysubrc.

  FIELD-SYMBOLS: <ls_enq>    TYPE seqg3.
  FIELD-SYMBOLS: <message>   TYPE ty_message.
  FIELD-SYMBOLS: <ls_status> TYPE ty_status.
  FIELD-SYMBOLS: <ls_object> TYPE rso_s_tlogo.
  FIELD-SYMBOLS: <ls_msg>    TYPE rs_s_msg.
  FIELD-SYMBOLS: <ls_tr_obj> TYPE trexreqob.


  CALL TRANSFORMATION (`ID`) SOURCE XML iv_xml_objects RESULT xml_objects = lt_objects.

  " only accept 1 - check / 2 - activate / 3 - transport mode
  ASSERT iv_mode = 1 OR iv_mode = 2 OR iv_mode = 3.

  CHECK lt_objects IS NOT INITIAL. " no objects -> nothing to do

  " create AWBN Factory for later usage
  CREATE OBJECT lo_awbn_factory.


  LOOP AT lt_objects ASSIGNING <ls_object>.

    " Prepare Status Output
    APPEND INITIAL LINE TO lt_status ASSIGNING <ls_status>.

    " Fill general fields
    <ls_status>-tlogo = <ls_object>-tlogo.
    <ls_status>-objnm = <ls_object>-objnm.

    " get logsys
    lv_logsys = cl_rso_repository=>get_logical_system_self( ).
    " get tlogoprops
    cl_rso_repository=>get_tlogoprop( EXPORTING i_tlogo = <ls_object>-tlogo IMPORTING
e_s_tlogoprop = ls_tlogoprop ).

    TRY .
        " Check A Version
```

```
        lv_objvers = 'A'.
        CALL METHOD cl_rso_object_collection_cache=>get_object_properties
          EXPORTING
            i_tlogo            = <ls_object>-tlogo
            i_objnm            = <ls_object>-objnm
            i_objvers          = lv_objvers
            i_objlogsys        = lv_logsys
            i_bypassing_buffer = abap_true
            i_no_message       = abap_true
          IMPORTING
            e_objstat          = lv_objstat
          EXCEPTIONS
            object_not_found   = 1.
      CATCH cx_root.
        " Check M Version
        lv_objvers = 'M'.
        CALL METHOD cl_rso_object_collection_cache=>get_object_properties
          EXPORTING
            i_tlogo            = <ls_object>-tlogo
            i_objnm            = <ls_object>-objnm
            i_objvers          = lv_objvers
            i_objlogsys        = lv_logsys
            i_bypassing_buffer = abap_true
            i_no_message       = abap_true
          IMPORTING
            e_objstat          = lv_objstat
          EXCEPTIONS
            object_not_found   = 1.
    ENDTRY.

    " fill Object Status
    <ls_status>-objstat = lv_objstat.

    " Check if tlogo class is available
    IF ls_tlogoprop-class IS NOT INITIAL.
      CALL METHOD (ls_tlogoprop-class)=>if_rso_tlogo~get_instance
        EXPORTING
          i_objnm   = <ls_object>-objnm
        RECEIVING
          r_r_tlogo = lo_tlogo.
    ELSE.
      <ls_status>-no_handler = abap_true.
    ENDIF.

    " saved?
    IF lo_tlogo IS BOUND.
      <ls_status>-saved = lo_tlogo->if_rso_tlogo_maintain~is_saved( ).
    ENDIF.

    IF iv_mode = '1'. " Check Mode

      " Check for M / A Version
      lo_awbn_factory->get_instance(
        EXPORTING
          i_tlogo              = <ls_object>-tlogo
          i_objnm              = <ls_object>-objnm
        RECEIVING
          re_r_awbobj          = lo_awbn_object ).

      <ls_status>-activflg = lo_awbn_object->get_activflg( ). " U -> M <> A Version / X -> M = A
Version / '' no information
      "Special case Analysis Auth.
      IF <ls_object>-tlogo = 'ASOB'.
        DATA: lt_objstat TYPE TABLE OF rsobjstat.
        SELECT objstat
          FROM rsecbiau
          INTO TABLE lt_objstat
            WHERE auth = <ls_object>-objnm.

        FIND 'INA' IN TABLE lt_objstat.

        IF sy-subrc = 0.
          <ls_status>-activflg = 'U'.
        ENDIF.
      "special case 7.x InfoSources
      ELSEIF <ls_object>-tlogo = 'TRCS'.
        DATA: lv_timstmp_a TYPE rstimestmp,
              lv_timstmp_m TYPE rstimestmp.
```

```
        SELECT SINGLE timestmp
          FROM rsksnew
            INTO lv_timstmp_a
              WHERE isource = <ls_object>-objnm
                AND objvers = 'A'.

        SELECT SINGLE timestmp
          FROM rsksnew
            INTO lv_timstmp_m
              WHERE isource = <ls_object>-objnm
                AND objvers = 'M'.

        IF lv_timstmp_m > lv_timstmp_a.
          <ls_status>-activflg = 'U'.
        ENDIF.
      ENDIF.

      " check for lock
      CALL FUNCTION 'ENQUEUE_READ'
        EXPORTING
          gclient = sy-mandt
          guname  = ''
        TABLES
          enq     = lt_enq.

      CONCATENATE <ls_object>-objnm '*' INTO lv_garg.

      LOOP AT lt_enq ASSIGNING <ls_enq> WHERE garg CP lv_garg. " AND gobj = ls_tlogoprop-
enqobject.
      ENDLOOP.
      IF sy-subrc = 0.
        <ls_status>-locked = abap_true.
      ENDIF.

      " transport locks
      DATA: ls_e071     TYPE e071.
      DATA: lv_result   TYPE trpari-s_checked.
      DATA: ls_tadir    TYPE tadir.
      DATA: ls_lock_key TYPE tlock_int.
      DATA: lt_tlock    TYPE TABLE OF tlock.
      DATA: lv_lockflag TYPE trpari-s_lockflag.

      ls_e071-pgmid    = 'R3TR'.
      ls_e071-object   = <ls_object>-tlogo.
      ls_e071-obj_name = <ls_object>-objnm.

      CALL FUNCTION 'TR_CHECK_TYPE'
        EXPORTING
          wi_e071     = ls_e071
        IMPORTING
          pe_result   = lv_result
          we_lock_key = ls_lock_key
          we_tadir    = ls_tadir.

      " determine lock key for complete object (TADIR)
      IF  ls_e071-pgmid    = ls_tadir-pgmid
      AND ls_e071-object   = ls_tadir-object
      AND ls_e071-obj_name = ls_tadir-obj_name.
        "   object is already complete (TADIR object): lock key is already known
      ELSE.
        MOVE-CORRESPONDING ls_tadir TO ls_e071.
        CALL FUNCTION 'TR_CHECK_TYPE'
          EXPORTING
            wi_e071     = ls_e071
          IMPORTING
            pe_result   = lv_result
            we_lock_key = ls_lock_key
            we_tadir    = ls_tadir.
      ENDIF.
      " proceed only for lockable objects
      IF lv_result = 'L'.
        " determine locks refering to complete (TADIR) object
        CALL FUNCTION 'TRINT_CHECK_LOCKS'
          EXPORTING
            wi_lock_key = ls_lock_key
          IMPORTING
            we_lockflag = lv_lockflag
          TABLES
```

```
            wt_tlock    = lt_tlock
          EXCEPTIONS
            empty_key   = 1
            OTHERS      = 2.
        IF lv_lockflag = abap_true.
          <ls_status>-tr_lock = abap_true.
        ELSE.
          <ls_status>-tr_lock = abap_false.
        ENDIF.

      ELSE.
        <ls_status>-tr_lock = abap_false.
      ENDIF.


      IF lo_tlogo IS BOUND.
        lo_tlogo->if_rso_tlogo_maintain~check(
          EXPORTING
            i_objvers            = lv_objvers
          IMPORTING
            e_r_msg              = lo_msg
            e_is_repaired        = lv_repaired
            e_subrc              = lv_subrc ).

        IF lo_msg IS BOUND.
          lt_msg = lo_msg->get_all_msg( ).

          LOOP AT lt_msg ASSIGNING <ls_msg>.
            APPEND INITIAL LINE TO <ls_status>-messages ASSIGNING <message>.
            <message>-message_type = <ls_msg>-msgty.
            MESSAGE ID <ls_msg>-msgid TYPE <ls_msg>-msgty NUMBER <ls_msg>-msgno
                 WITH <ls_msg>-msgv1 <ls_msg>-msgv2 <ls_msg>-msgv3 <ls_msg>-msgv4
                 INTO <message>-message.
          ENDLOOP.
        ENDIF.
      ENDIF.

    ENDIF.

    IF iv_mode = '2'. " Activation Mode

      IF lo_tlogo IS NOT BOUND.
        EXIT.
      ENDIF.

      " Run Check before activation.
      IF lo_tlogo IS BOUND.
        lo_tlogo->if_rso_tlogo_maintain~check(
      EXPORTING
        i_objvers            = lv_objvers
      IMPORTING
        e_r_msg              = lo_msg
        e_is_repaired        = lv_repaired
        e_subrc              = lv_subrc ).
        IF lo_msg IS BOUND.
          lt_msg = lo_msg->get_all_msg( ).

          DELETE lt_msg WHERE NOT ( msgty EQ 'E' OR msgty EQ 'W' ). " only look for errors and
warnings
          IF lt_msg IS NOT INITIAL. " Error messages found? return messages - skip activation
            LOOP AT lt_msg ASSIGNING <ls_msg>.
              APPEND INITIAL LINE TO <ls_status>-messages ASSIGNING <message>.
              <message>-message_type = <ls_msg>-msgty.
              MESSAGE ID <ls_msg>-msgid TYPE <ls_msg>-msgty NUMBER <ls_msg>-msgno
                   WITH <ls_msg>-msgv1 <ls_msg>-msgv2 <ls_msg>-msgv3 <ls_msg>-msgv4
                   INTO <message>-message.
            ENDLOOP.
          ENDIF.
        ENDIF.
      ENDIF.
      TRY.
          lo_tlogo->if_rso_tlogo_maintain~prepare(
            EXPORTING
              i_with_enqueue       = rs_c_false    " = 'X': with Enqueue Lock
              i_with_cto_check     = rs_c_false    " = 'X': with CTO Check
              i_with_authority     = rs_c_false    " = 'X': with Authorization
              i_with_rebuild       = rs_c_false    " = 'X': Database Must Reread the Object
              i_actvt              = rssb_c_auth_actvt-maintain    " Activity
```

```
              i_objvers            = rs_c_objvers-active    " Object version
              i_detlevel           = '3'    " Application Log: Level of Detail
          ).
        CATCH cx_rs_cancelled.
        CATCH cx_rs_not_authorized.
        CATCH cx_rs_display_only.
      ENDTRY.

      lo_tlogo->if_rso_tlogo_maintain~activate(
        EXPORTING
          i_objvers             = lv_objvers   " Object Version (A / M)
          i_force_activation    = rs_c_true    " = 'X': activate in case the object is already
active
          i_show_check_protocol = rs_c_false   " = 'X': Display Consistency Log as Popup if
Warnings Arise
          i_with_cto            = rs_c_false
        IMPORTING
          e_subrc               =  lv_subrc  ).

      IF lv_subrc = 0.
        <ls_status>-activation_successful = abap_true.
        APPEND INITIAL LINE TO <ls_status>-messages ASSIGNING <message>.
        <message>-message_type = 'S'.
        <message>-message = 'Activation successful'.
      ELSE.
        <ls_status>-activation_successful = abap_false.
        APPEND INITIAL LINE TO <ls_status>-messages ASSIGNING <message>.
        <message>-message_type = 'E'.
        <message>-message = 'Activation failed'.
      ENDIF.

    ENDIF.

    IF iv_mode = '3'. " Transport Mode
      CHECK iv_transport IS NOT INITIAL.

      DATA: ls_wi_e071 LIKE e071.
      ls_wi_e071-pgmid = 'R3TR'.
      ls_wi_e071-object   = <ls_object>-tlogo.
      ls_wi_e071-obj_name = <ls_object>-objnm.

      CALL FUNCTION 'TR_APPEND_TO_COMM'
        EXPORTING
          pi_korrnum                    = iv_transport
          wi_e071                       = ls_wi_e071
        EXCEPTIONS
          no_authorization              = 1
          no_systemname                 = 2
          no_systemtype                 = 3
          tr_check_keysyntax_error      = 4
          tr_check_obj_error            = 5
          tr_enqueue_failed             = 6
          tr_ill_korrnum                = 7
          tr_key_without_header         = 8
          tr_lockmod_failed             = 9
          tr_lock_enqueue_failed        = 10
          tr_modif_only_in_modif_order  = 11
          tr_not_owner                  = 12
          tr_no_append_of_corr_entry    = 13
          tr_no_append_of_c_member      = 14
          tr_no_shared_repairs          = 15
          tr_order_not_exist            = 16
          tr_order_released             = 17
          tr_order_update_error         = 18
          tr_repair_only_in_repair_order = 19
          tr_wrong_order_type           = 20
          wrong_client                  = 21
          OTHERS                        = 22.
      IF sy-subrc <> 0.
        <ls_status>-written_to_tr = abap_false.
      ELSE.
        <ls_status>-written_to_tr = abap_true.
      ENDIF.
      FREE: ls_wi_e071.

    ENDIF.

  ENDLOOP.
```

```
  CALL TRANSFORMATION (`ID`) SOURCE status   = lt_status   RESULT XML ev_xml_status.

ENDFUNCTION.
```

6 Function Module Z_RFC_CODESCAN - ABAP source scan (RFC)

6.1 General Information

System	BI2
Function Module	Z_RFC_CODESCAN, ABAP source scan (RFC)
Function Pool	Z_DP
Remote	Yes
Last changed by	AKOLMOG
Last change (timestamp)	18/09/2019
Timestamp of documentation	11/08/2020 16:29:36

6.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
I_CASE	TYPE	RS_BOOL	''	X	X	
I_SKIP_COMMENT	TYPE	RS_BOOL	'X'	X	X	
I_REGEX	TYPE	RS_BOOL	''	X	X	
I_UPDRU	TYPE	RS_BOOL	'X'	X	X	
I_TRANR	TYPE	RS_BOOL	'X'	X	X	
I_TRANS	TYPE	RS_BOOL	'X'	X	X	
I_IOBJROUT	TYPE	RS_BOOL	'X'	X	X	
I_IP	TYPE	RS_BOOL		X	X	
I_DTP	TYPE	RS_BOOL		X	X	
I_REPS	TYPE	RS_BOOL		X	X	
I_CLAS	TYPE	RS_BOOL		X	X	
I_FUNC	TYPE	RS_BOOL		X	X	
I_PLSE	TYPE	RS_BOOL		X	X	

6.3 Tables

Parameter		Associated Type	Opt.	Short text
T_STRING_RANGE	LIKE	RSRANGE		
T_DEVCL_RANGE	LIKE	RSRANGE		
T_REPS_RANGE	LIKE	RSRANGE		
T_SUBC_RANGE	LIKE	RSRANGE		
T_CLAS_RANGE	LIKE	RSRANGE		
T_FUGR_RANGE	LIKE	RSRANGE		
DATA	LIKE	TAB512		

6.4 Exceptions

Exception	Short text
RELEASE_2_1	
INPUT_CANNOT_BE_TREATED	

6.5 Source code Function module

```
FUNCTION Z_RFC_CODESCAN.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_CASE) TYPE  RS_BOOL DEFAULT ''
```

```
*"     VALUE(I_SKIP_COMMENT) TYPE  RS_BOOL DEFAULT 'X'
*"     VALUE(I_REGEX) TYPE  RS_BOOL DEFAULT ''
*"     VALUE(I_UPDRU) TYPE  RS_BOOL DEFAULT 'X'
*"     VALUE(I_TRANR) TYPE  RS_BOOL DEFAULT 'X'
*"     VALUE(I_TRANS) TYPE  RS_BOOL DEFAULT 'X'
*"     VALUE(I_IOBJROUT) TYPE  RS_BOOL DEFAULT 'X'
*"     VALUE(I_IP) TYPE  RS_BOOL OPTIONAL
*"     VALUE(I_DTP) TYPE  RS_BOOL OPTIONAL
*"     VALUE(I_REPS) TYPE  RS_BOOL OPTIONAL
*"     VALUE(I_CLAS) TYPE  RS_BOOL OPTIONAL
*"     VALUE(I_FUNC) TYPE  RS_BOOL OPTIONAL
*"     VALUE(I_PLSE) TYPE  RS_BOOL OPTIONAL
*"  TABLES
*"      T_STRING_RANGE STRUCTURE  RSRANGE
*"      T_DEVCL_RANGE STRUCTURE  RSRANGE
*"      T_REPS_RANGE STRUCTURE  RSRANGE
*"      T_SUBC_RANGE STRUCTURE  RSRANGE
*"      T_CLAS_RANGE STRUCTURE  RSRANGE
*"      T_FUGR_RANGE STRUCTURE  RSRANGE
*"      DATA STRUCTURE  TAB512
*"  EXCEPTIONS
*"      RELEASE_2_1
*"      INPUT_CANNOT_BE_TREATED
*"----------------------------------------------------------------------

  DATA:
    ls_data             LIKE LINE OF data,
    ls_result           TYPE s_result,
    ls_line             TYPE c LENGTH 255,
    lv_nspacegen        TYPE namespace,
    lv_name_w_o_prefix TYPE rs_char30,
    lt_string           TYPE t_string,
    ls_string           TYPE s_string.

  FIELD-SYMBOLS:
    <string>  TYPE s_string,
    <s_range> TYPE rsrange.

*---------------------------------------------------------------------

  CLEAR:
    gt_rsaabap,
    gt_object,
    gt_result.

  LOOP AT t_string_range ASSIGNING <s_range>.

    IF ( <s_range>-high = 'ODSO' OR <s_range>-high = 'CUBE'
      OR <s_range>-high = 'CHAR' OR <s_range>-high = 'IOBJ')
      OR <s_range>-high = 'ADSO'. "ADü_20189424 - DP-96
      CALL FUNCTION 'RSD_NSPACE_PAR_GET_FROM_NAME'
        EXPORTING
          i_objnm           = <s_range>-low
        IMPORTING
*         E_NAMESPACE       =
          e_nspacegen       = lv_nspacegen
*         E_SYSTP           =
          e_name_w_o_prefix = lv_name_w_o_prefix
*         E_SOBJNM_W_O_PREFIX =
        EXCEPTIONS
          name_error        = 1
          OTHERS            = 2.

      IF sy-subrc = 0.
        "regular expressions: the . is a placeholder for a single sign
        IF <s_range>-high = 'ODSO'.
          "sign after prefix is A
          CONCATENATE lv_nspacegen 'A' lv_name_w_o_prefix '.0'
           INTO ls_string.
          "Start Change ADü_20189424 - DP-96
        ELSEIF <s_range>-high = 'ADSO'.
          "sign after prefix is A
          CONCATENATE lv_nspacegen 'A' lv_name_w_o_prefix '.'
           INTO ls_string.
          "End Change ADü_20189424 - DP-96
        ELSEIF <s_range>-high = 'CUBE'.
          "sign after prefix can be D,E or F
          CONCATENATE lv_nspacegen '.' lv_name_w_o_prefix
```

```
          INTO ls_string.
      ELSEIF ( <s_range>-high = 'CHAR' OR <s_range>-high = 'IOBJ' ).
        CONCATENATE lv_nspacegen '.' lv_name_w_o_prefix
         INTO ls_string.

        "Start Change ADü_20180426 - DP-1945
        DATA: ls_string2 TYPE s_string,
              ls_string3 TYPE s_string,
              ls_fieldnm TYPE rsdiobjfieldnm.

        SELECT SINGLE fieldnm
          FROM rsdiobj
            INTO ls_fieldnm
              WHERE iobjnm  = <s_range>-low
                AND objvers = 'A'.

        IF ls_fieldnm IS NOT INITIAL.

          CONCATENATE '-' ls_fieldnm '\.' INTO ls_string2.
          APPEND ls_string2 TO lt_string.
          CONCATENATE '-' ls_fieldnm '\s' INTO ls_string3.
          APPEND ls_string3 TO lt_string.

        ENDIF.
        "End Change ADü_20180426 - DP-1945

      ENDIF.
    ENDIF.
  ELSE.
    ls_string = <s_range>-low.
  ENDIF.

  APPEND ls_string TO lt_string.

ENDLOOP.


IF i_updru = 'X' OR i_tranr = 'X' OR i_trans = 'X'
   OR i_iobjrout = 'X'.
  PERFORM scan_rsaabap
    TABLES lt_string
    USING i_skip_comment i_case i_regex i_updru
          i_tranr i_trans i_iobjrout.
ENDIF.

IF i_dtp = 'X'.
  PERFORM scan_dtp
    TABLES lt_string t_devcl_range
    USING i_skip_comment i_case i_regex.
ENDIF.

IF i_ip = 'X'.
  PERFORM scan_ip
    TABLES lt_string t_devcl_range
    USING i_skip_comment i_case i_regex.
ENDIF.

IF i_reps = 'X'.
  PERFORM get_reps
    TABLES t_devcl_range t_reps_range t_subc_range.
ENDIF.

IF i_clas = 'X'.
  PERFORM get_clas
    TABLES t_devcl_range t_clas_range.
ENDIF.

IF i_func = 'X'.
  PERFORM get_func
  TABLES t_devcl_range t_fugr_range.
ENDIF.

IF i_plse = 'X'.
  PERFORM get_plse
  TABLES lt_string t_devcl_range
    USING i_skip_comment i_case i_regex.
ENDIF.
```

```
  IF NOT gt_object IS INITIAL.
    PERFORM scan_include_code
      TABLES lt_string
        USING i_skip_comment i_case i_regex.
  ENDIF.

  "Fill result table
  LOOP AT gt_result INTO ls_result.
    IF i_skip_comment = 'X'.
      IF ls_result-line(1) = '*'.
        CONTINUE.
      ELSE.
        ls_line = ls_result-line.
        SHIFT ls_line LEFT DELETING LEADING space.
        "Start change ADü_20180618 - DP-1210
        "AMDP Code
        IF   ls_result-routinetype = 'RT20'
          OR ls_result-routinetype = 'RT21'
          OR ls_result-routinetype = 'RT22'
          OR ls_result-routinetype = 'RT23'
          OR ls_result-routinetype = 'RT24'
          OR ls_result-routinetype = 'RT30'
          OR ls_result-routinetype = 'RT31'
          OR ls_result-routinetype = 'RT32'
          OR ls_result-routinetype = 'RT33'.
          IF ls_line(2) = '--'.
            CONTINUE.
          ENDIF.
          "ABAP Code
        ELSE.
          "End change ADü_20180618 - DP-1210
          IF ls_line(1) = '"'.
            CONTINUE.
          ENDIF.
          "Start change ADü_20180618 - DP-1210
        ENDIF.
        "End change ADü_20180618 - DP-1210
      ENDIF.
    ENDIF.
    ls_data = ls_result.
    APPEND ls_data TO data.

  ENDLOOP.

ENDFUNCTION.


*&---------------------------------------------------------------------*
*&      Form  scan_rsaabap
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM scan_rsaabap
    TABLES lt_string
    USING i_skip_comment i_case i_regex i_updru i_tranr
          i_trans i_iobjrout.

  TYPES:
    BEGIN OF s_lookup,
      targetname    LIKE rstran-targetname,
      targettype    LIKE rstran-targettype,
      targetsubtype LIKE rstran-targetsubtype,
      sourcename    LIKE rstran-sourcename,
      sourcetype    LIKE rstran-sourcetype,
      sourcesubtype LIKE rstran-sourcesubtype,
      codeid        LIKE rsaabap-codeid,
      tranid        LIKE rstran-tranid,
      set_runtime   TYPE char1,
      iobjnm        LIKE rsupdrout-iciobjnm,
      routinetype   TYPE c LENGTH 4,
      routine       TYPE n LENGTH 4,
    END OF s_lookup,
    t_lookup TYPE STANDARD TABLE OF s_lookup.

  DATA:
    l_tabix          LIKE sy-tabix,
    lt_rsaabap       TYPE t_rsaabap,
```

```
    ls_rsaabap       TYPE s_rsaabap,
    ls_match_result  TYPE match_result,
    lt_lookup_finder TYPE t_lookup,
    ls_lookup_temp   TYPE s_lookup,
    lt_lookup_temp   TYPE t_lookup,
    ls_lookup        TYPE s_lookup,
    l_updtype        LIKE rsupddat-updtype,
    ls_result        TYPE s_result,
    lv_txtlg         TYPE rstxtlg.

  FIELD-SYMBOLS:
    <string>  TYPE s_string,
    <s_range> TYPE rsrange.


  CLEAR gt_rsaabap.
* Restrict on version M, because RSAABAP contains deleted coding
* that exists only in version A but not in version M!
  SELECT a~codeid codetp line_no line
    FROM              rsaabap  AS a
      INNER JOIN      rsarout  AS r
    ON a~codeid = r~codeid AND a~objvers = r~objvers
    INTO CORRESPONDING FIELDS OF TABLE gt_rsaabap
    WHERE a~objvers = 'A'
    AND a~codeid IN (
      SELECT codeid FROM rsarout
        WHERE objvers = 'M').                                "#EC *

  "Start Change ADü_20180618 - DP-1210
  DATA: lv_rstranscript     TYPE string VALUE 'RSTRANSCRIPT',
        lv_rstranstepscript TYPE string VALUE 'RSTRANSTEPSCRIPT',
        lv_rstransteprout   TYPE string VALUE 'RSTRANSTEPROUT',
        lv_bwrelease        TYPE saprelease,
        lv_bw4hana          TYPE rs_bool VALUE ''.

  TYPES: BEGIN OF lty_sscript,
           tranid  TYPE rstranid,
           script  TYPE string,
           ruleid  TYPE rstran_ruleid,
           stepid  TYPE rstran_stepid,
           codeid  TYPE rscodeid,
           line_no TYPE rsaabap-line_no,
         END OF lty_sscript,

         BEGIN OF lty_script,
           tranid  TYPE rstranid,
           procnm  TYPE char30,
           codeid  TYPE rscodeid,
           script  TYPE string,
           line_no TYPE rsaabap-line_no,
         END OF lty_script,

         BEGIN OF lty_rstransteprout,
           tranid   TYPE rstranid,
           ruleid   TYPE rstran_ruleid,
           stepid   TYPE rstran_stepid,
           codeid   TYPE rscodeid,
           on_hana  TYPE rs_bool,
           line_no  TYPE rsaabap-line_no,
           kind     TYPE char7,
           code     TYPE string,
           code_inv TYPE string,
         END OF lty_rstransteprout.

  DATA: ls_script           TYPE lty_script,
        ls_g_script         TYPE lty_script,
        ls_sscript          TYPE lty_sscript,
        ls_g_sscript        TYPE lty_sscript,
        lt_script           TYPE TABLE OF lty_script,
        lt_g_script         TYPE TABLE OF lty_script,
        lt_sscript          TYPE TABLE OF lty_sscript,
        lt_g_sscript        TYPE TABLE OF lty_sscript,
        ls_rstransteprout   TYPE lty_rstransteprout,
        ls_g_rstransteprout TYPE lty_rstransteprout,
        lt_rstransteprout   TYPE TABLE OF lty_rstransteprout,
        lt_g_rstransteprout TYPE TABLE OF lty_rstransteprout..

  SELECT SINGLE release
```

```
  FROM cvers
    INTO lv_bwrelease
    WHERE component = 'SAP_BW'.

IF lv_bwrelease IS INITIAL.
  lv_bw4hana = rs_c_true.
  SELECT SINGLE release
  FROM cvers
    INTO lv_bwrelease
    WHERE component = 'DW4CORE'.
ENDIF.

IF lv_bw4hana = rs_c_false.
  IF lv_bwrelease < 740.
    "nothing
  ELSEIF lv_bwrelease >= 740 AND lv_bwrelease < 750.

    SELECT tranid procnm script
      FROM (lv_rstranscript)
        INTO CORRESPONDING FIELDS OF TABLE lt_script
          WHERE objvers = 'A'.

  ELSEIF lv_bwrelease >= 750.

    SELECT tranid procnm script
          FROM (lv_rstranscript)
            INTO CORRESPONDING FIELDS OF TABLE lt_script
              WHERE objvers = 'A'.

    SELECT tranid ruleid stepid codeid script
          FROM (lv_rstranstepscript)
            INTO CORRESPONDING FIELDS OF TABLE lt_sscript
              WHERE objvers = 'A'.
  ENDIF.
ELSEIF lv_bw4hana = rs_c_true.
  SELECT tranid ruleid stepid codeid code code_inv
    FROM (lv_rstransteprout)
      INTO CORRESPONDING FIELDS OF TABLE lt_rstransteprout
        WHERE objvers = 'A'.
ENDIF.
"End Change ADü_20180618 - DP-1210

"Start Change ADü_20180426 - DP-1945
DATA: ls_complstring TYPE string.
LOOP AT lt_string ASSIGNING <string>.
  IF sy-tabix = 1.
    ls_complstring = <string>.
  ELSE.
    CONCATENATE ls_complstring '|' <string> INTO ls_complstring.
  ENDIF.
ENDLOOP.
"End Change ADü_20180426 - DP-1945

SORT gt_rsaabap BY codeid.
LOOP AT gt_rsaabap INTO ls_rsaabap.
  l_tabix = sy-tabix.
  "LOOP AT lt_string ASSIGNING <string>. "ADü_20180426 - DP-1945
  CLEAR ls_match_result.
  IF i_case = ' ' AND i_regex = 'X'.
    FIND FIRST OCCURRENCE OF REGEX ls_complstring "<string> "ADü_20180426 - DP-1945
    IN ls_rsaabap-line
      IN CHARACTER MODE
      IGNORING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = 'X' AND i_regex = 'X'.
    FIND FIRST OCCURRENCE OF REGEX ls_complstring "<string> "ADü_20180426 - DP-1945
    IN ls_rsaabap-line
      IN CHARACTER MODE
      RESPECTING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = ' ' AND i_regex = ''.
    FIND FIRST OCCURRENCE OF ls_complstring "<string> "ADü_20180426 - DP-1945
    IN ls_rsaabap-line
      IN CHARACTER MODE
      IGNORING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = 'X' AND i_regex = ''.
    FIND FIRST OCCURRENCE OF ls_complstring "<string> "ADü_20180426 - DP-1945
```

```
    IN ls_rsaabap-line
      IN CHARACTER MODE
      RESPECTING CASE
      RESULTS ls_match_result.
  ENDIF.
  IF sy-subrc <> 0.
    DELETE gt_rsaabap INDEX l_tabix.
  ENDIF.
  "ENDLOOP. "ADü_20180426 - DP-1945
ENDLOOP.

"Start Change ADü_20180618 - DP-1210
"get the full scripts with comments:
CLASS cl_oo_include_naming DEFINITION LOAD.
DATA lo_clif_incl_naming  TYPE REF TO if_oo_clif_incl_naming.
DATA lo_class_incl_naming TYPE REF TO if_oo_class_incl_naming.
DATA l_t_method_w_include TYPE seop_methods_w_include.
FIELD-SYMBOLS: <s_method_w_include> TYPE seop_method_w_include.
DATA: l_clskey TYPE seoclskey,
      lt_abap  TYPE abaptxt255_tab,
      ls_abap  TYPE LINE OF abaptxt255_tab,
      length   TYPE i,
      last     TYPE char30.

"HANA Script
LOOP AT lt_script INTO ls_script.

  CONCATENATE '/BIC/' ls_script-procnm+3 INTO l_clskey.

  CALL METHOD cl_oo_include_naming=>get_instance_by_cifkey
    EXPORTING
      cifkey         = l_clskey
    RECEIVING
      cifref         = lo_clif_incl_naming
    EXCEPTIONS
      no_objecttype  = 1
      internal_error = 2
      OTHERS         = 3.

  IF sy-subrc = 0.
    lo_class_incl_naming ?= lo_clif_incl_naming.
    l_t_method_w_include =
      lo_class_incl_naming->get_all_method_includes( ).
    LOOP AT l_t_method_w_include ASSIGNING <s_method_w_include>.
      CLEAR: length, last.
      length = strlen( <s_method_w_include>-cpdkey ) - 9.
      last = <s_method_w_include>-cpdkey+length.
      IF last = 'PROCEDURE'.
        CLEAR lt_abap.
        READ REPORT <s_method_w_include>-incname INTO lt_abap.
        LOOP AT lt_abap INTO ls_abap.
          ls_g_script-tranid   = ls_script-tranid.
          ls_g_script-procnm   = ls_script-procnm.
          ls_g_script-codeid   = ls_script-procnm+3.
          ls_g_script-script   = ls_abap.
          ls_g_script-line_no  = sy-tabix.
          APPEND ls_g_script TO lt_g_script.
        ENDLOOP.
      ENDIF.
    ENDLOOP.
  ENDIF.
ENDLOOP.

"AMDP
LOOP AT lt_sscript INTO ls_sscript.

  CONCATENATE '/BIC/' ls_sscript-codeid INTO l_clskey.

  CALL METHOD cl_oo_include_naming=>get_instance_by_cifkey
    EXPORTING
      cifkey         = l_clskey
    RECEIVING
      cifref         = lo_clif_incl_naming
    EXCEPTIONS
      no_objecttype  = 1
      internal_error = 2
      OTHERS         = 3.
```

```
  IF sy-subrc = 0.
    lo_class_incl_naming ?= lo_clif_incl_naming.
    l_t_method_w_include =
      lo_class_incl_naming->get_all_method_includes( ).
    LOOP AT l_t_method_w_include ASSIGNING <s_method_w_include>.
      CLEAR: length, last.
      length = strlen( <s_method_w_include>-cpdkey ) - 9.
      last = <s_method_w_include>-cpdkey+length.
      IF last = 'PROCEDURE'.
        CLEAR lt_abap.
        READ REPORT <s_method_w_include>-incname INTO lt_abap.
        LOOP AT lt_abap INTO ls_abap.
          ls_g_sscript-tranid   = ls_sscript-tranid.
          ls_g_sscript-codeid   = ls_sscript-codeid.
          ls_g_sscript-script   = ls_abap.
          ls_g_sscript-line_no  = sy-tabix.
          APPEND ls_g_sscript TO lt_g_sscript.
        ENDLOOP.
      ENDIF.
    ENDLOOP.
  ENDIF.
ENDLOOP.

"BW4HANA
LOOP AT lt_rstransteprout INTO ls_rstransteprout.
  CLEAR lt_abap.
  SPLIT ls_rstransteprout-code AT cl_abap_char_utilities=>newline INTO TABLE lt_abap.
  LOOP AT lt_abap INTO ls_abap.
    ls_g_rstransteprout-tranid   = ls_rstransteprout-tranid.
    ls_g_rstransteprout-codeid   = ls_rstransteprout-codeid.
    ls_g_rstransteprout-code     = ls_abap.
    ls_g_rstransteprout-line_no  = sy-tabix.
    APPEND ls_g_rstransteprout TO lt_g_rstransteprout.
  ENDLOOP.
ENDLOOP.

"Search code in new tables:
"RSTRANSCRIPT
LOOP AT lt_g_script INTO ls_g_script.
  l_tabix = sy-tabix.
  CLEAR ls_match_result.
  IF i_case = ' ' AND i_regex = 'X'.
    FIND FIRST OCCURRENCE OF REGEX ls_complstring
    IN ls_g_script-script
      IN CHARACTER MODE
      IGNORING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = 'X' AND i_regex = 'X'.
    FIND FIRST OCCURRENCE OF REGEX ls_complstring
    IN ls_g_script-script
      IN CHARACTER MODE
      RESPECTING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = ' ' AND i_regex = ''.
    FIND FIRST OCCURRENCE OF ls_complstring
    IN ls_g_script-script
      IN CHARACTER MODE
      IGNORING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = 'X' AND i_regex = ''.
    FIND FIRST OCCURRENCE OF ls_complstring
    IN ls_g_script-script
      IN CHARACTER MODE
      RESPECTING CASE
      RESULTS ls_match_result.
  ENDIF.
  IF sy-subrc <> 0.
    DELETE lt_g_script INDEX l_tabix.
  ENDIF.
ENDLOOP.

"RSTRANSTEPSCRIPT
LOOP AT lt_g_sscript INTO ls_g_sscript.
  l_tabix = sy-tabix.
  CLEAR ls_match_result.
  IF i_case = ' ' AND i_regex = 'X'.
    FIND FIRST OCCURRENCE OF REGEX ls_complstring
    IN ls_g_sscript-script
```

```
      IN CHARACTER MODE
      IGNORING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = 'X' AND i_regex = 'X'.
    FIND FIRST OCCURRENCE OF REGEX ls_complstring
    IN ls_g_sscript-script
      IN CHARACTER MODE
      RESPECTING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = ' ' AND i_regex = ''.
    FIND FIRST OCCURRENCE OF ls_complstring
    IN ls_g_sscript-script
      IN CHARACTER MODE
      IGNORING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = 'X' AND i_regex = ''.
    FIND FIRST OCCURRENCE OF ls_complstring
    IN ls_g_sscript-script
      IN CHARACTER MODE
      RESPECTING CASE
      RESULTS ls_match_result.
  ENDIF.
  IF sy-subrc <> 0.
    DELETE lt_g_sscript INDEX l_tabix.
  ENDIF.
ENDLOOP.
"End Change ADü_20180618 - DP-1210

"RSTRANSTEPROUT
SORT lt_g_rstransteprout BY codeid.
LOOP AT lt_g_rstransteprout INTO ls_g_rstransteprout.
  l_tabix = sy-tabix.
  CLEAR ls_match_result.
  IF i_case = ' ' AND i_regex = 'X'.
    FIND FIRST OCCURRENCE OF REGEX ls_complstring
    IN ls_g_rstransteprout-code
      IN CHARACTER MODE
      IGNORING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = 'X' AND i_regex = 'X'.
    FIND FIRST OCCURRENCE OF REGEX ls_complstring
    IN ls_g_rstransteprout-code
      IN CHARACTER MODE
      RESPECTING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = ' ' AND i_regex = ''.
    FIND FIRST OCCURRENCE OF ls_complstring
    IN ls_g_rstransteprout-code
      IN CHARACTER MODE
      IGNORING CASE
      RESULTS ls_match_result.
  ELSEIF i_case = 'X' AND i_regex = ''.
    FIND FIRST OCCURRENCE OF ls_complstring
    IN ls_g_rstransteprout-code
      IN CHARACTER MODE
      RESPECTING CASE
      RESULTS ls_match_result.
  ENDIF.
  IF sy-subrc <> 0.
    DELETE lt_g_rstransteprout  INDEX l_tabix.
  ENDIF.
ENDLOOP.


IF gt_rsaabap IS INITIAL.
  "Start Change ADü_20180618 - DP-1210
  IF lt_g_script IS INITIAL AND lt_g_sscript IS INITIAL.
    "End Change ADü_20180618 - DP-1210
    IF lt_g_rstransteprout IS INITIAL.
      RETURN.
    ENDIF.
    "Start Change ADü_20180618 - DP-1210
  ENDIF.
  "End Change ADü_20180618 - DP-1210
ENDIF.

"---------------------------------------------------------------------
" Scan routines of Update Rules (3.x)
```

```
"---------------------------------------------------------------------
IF i_updru = 'X'.

  TYPES:
    BEGIN OF s_rsupd,
      updid    LIKE rsupdinfo-updid,
      infocube LIKE rsupdinfo-infocube,
      isource  LIKE rsupdinfo-isource,
      codeid   LIKE rsupdrout-codeid,
      routine  LIKE rsupdrout-routine,
      iciobjnm LIKE rsupdrout-iciobjnm,
    END OF s_rsupd,
    t_rsupd TYPE STANDARD TABLE OF s_rsupd.

  DATA: lt_rsupd TYPE t_rsupd.

  FIELD-SYMBOLS <s_rsupd> TYPE s_rsupd.

  CLEAR lt_rsaabap.
  LOOP AT gt_rsaabap INTO ls_rsaabap WHERE codetp = 'UR'.
    APPEND ls_rsaabap TO lt_rsaabap.
  ENDLOOP.
  IF NOT lt_rsaabap IS INITIAL.

    CLEAR lt_lookup_temp.
    SELECT a~updid infocube isource codeid routine iciobjnm
      INTO TABLE lt_rsupd
      FROM rsupdinfo AS a
      INNER JOIN rsupdrout AS b ON a~updid = b~updid
      FOR ALL ENTRIES IN lt_rsaabap
        WHERE
          b~codeid  = lt_rsaabap-codeid AND
          a~objvers = 'A'               AND
          b~objvers = 'A'.

    LOOP AT lt_rsupd ASSIGNING <s_rsupd>.
      ls_lookup_temp-targetname = <s_rsupd>-infocube.
      ls_lookup_temp-sourcename = <s_rsupd>-isource.
      ls_lookup_temp-codeid = <s_rsupd>-codeid.
      IF <s_rsupd>-routine = '9998'.
        ls_lookup_temp-routinetype = 'RT07'.
      ELSEIF <s_rsupd>-routine = '9999'.
        ls_lookup_temp-routinetype = 'RT09'.
      ELSE.
        ls_lookup_temp-routinetype = 'RT08'.
      ENDIF.
      ls_lookup_temp-routine = <s_rsupd>-routine.
      ls_lookup_temp-tranid = <s_rsupd>-updid.
      ls_lookup_temp-iobjnm = <s_rsupd>-iciobjnm.
      APPEND ls_lookup_temp TO lt_lookup_temp.
    ENDLOOP.

    SORT lt_lookup_temp BY codeid.
    LOOP AT lt_rsaabap INTO ls_rsaabap.

      CLEAR ls_lookup.
      READ TABLE lt_lookup_temp INTO ls_lookup
      WITH KEY codeid = ls_rsaabap-codeid BINARY SEARCH.
      IF sy-subrc = 0.
        CLEAR l_updtype.
        SELECT SINGLE updtype INTO l_updtype FROM rsupddat
          WHERE
            updid   = ls_lookup-tranid AND
            objvers = 'A'              AND
            routine = ls_lookup-routine.                  "#EC *

        "no update
        IF l_updtype = 'NOP'.
          DELETE lt_rsaabap.
        ELSE.
          CLEAR ls_result.
          MOVE-CORRESPONDING ls_lookup  TO ls_result.
          MOVE-CORRESPONDING ls_rsaabap TO ls_result.
          "Derive description of the Update Rule
          CONCATENATE ls_lookup-targetname ls_lookup-sourcename
                      INTO ls_result-txtlg SEPARATED BY space.
          APPEND ls_result TO gt_result.
        ENDIF.
```

```
      ENDIF.

    ENDLOOP.

  ENDIF.
ENDIF.

"---------------------------------------------------------------------
" Scan routines of Transfer Rules (3.x)
"---------------------------------------------------------------------
IF i_tranr = 'X'.

  TYPES:
    BEGIN OF ty_rstsrules,"relevant for InfoObject Routines
      transtru     LIKE rstsrules-transtru,
      comstru      LIKE rstsrules-comstru,
      iobjnm       LIKE rstsrules-iobjnm,
      convrout_l   LIKE rstsrules-convrout_l,
      startroutine LIKE rsts-startroutine,
    END OF ty_rstsrules,
    t_rstsrules TYPE STANDARD TABLE OF ty_rstsrules.

  DATA lt_rstsrules TYPE t_rstsrules.
  FIELD-SYMBOLS <s_rstsrules> TYPE ty_rstsrules.

  TYPES:
    BEGIN OF ty_rsts,"relevant for Startroutine and Global code
      transtru     LIKE rsts-transtru,
      odsname      LIKE rsts-odsname,
      glbcode      LIKE rsts-glbcode,
      startroutine LIKE rsts-startroutine,
    END OF ty_rsts,
    t_rsts TYPE STANDARD TABLE OF ty_rsts.

  DATA lt_rsts TYPE t_rsts.
  FIELD-SYMBOLS <s_rsts> TYPE ty_rsts.

  CLEAR lt_rsaabap.
  LOOP AT gt_rsaabap INTO ls_rsaabap WHERE codetp = 'TR'.
    APPEND ls_rsaabap TO lt_rsaabap.
  ENDLOOP.
  IF NOT lt_rsaabap IS INITIAL.

    CLEAR lt_lookup_temp.
    SELECT transtru comstru iobjnm convrout_l
      INTO TABLE lt_rstsrules
      FROM rstsrules
      FOR ALL ENTRIES IN lt_rsaabap
      WHERE
        convrout_l = lt_rsaabap-codeid AND
        objvers    = 'A'.                                    "#EC *

    SELECT transtru odsname glbcode startroutine
    INTO TABLE lt_rsts
    FROM rsts
    FOR ALL ENTRIES IN lt_rsaabap
    WHERE
      ( glbcode = lt_rsaabap-codeid OR
      startroutine = lt_rsaabap-codeid )
      AND
      objvers    = 'A'.

    CLEAR ls_lookup_temp.
    LOOP AT lt_rstsrules ASSIGNING <s_rstsrules>.
      ls_lookup_temp-tranid = <s_rstsrules>-transtru.
      ls_lookup_temp-codeid = <s_rstsrules>-convrout_l.
      ls_lookup_temp-iobjnm = <s_rstsrules>-iobjnm.
      APPEND ls_lookup_temp TO lt_lookup_temp.
    ENDLOOP.

    CLEAR ls_lookup_temp.
    LOOP AT lt_rsts ASSIGNING <s_rsts>.
      ls_lookup_temp-tranid = <s_rsts>-transtru.
      ls_lookup_temp-iobjnm = ''.
      IF <s_rsts>-glbcode IS NOT INITIAL.
        ls_lookup_temp-codeid = <s_rsts>-glbcode.
        ls_lookup_temp-routinetype = 'RT12'.
        APPEND ls_lookup_temp TO lt_lookup_temp.
```

```
      ENDIF.
      IF <s_rsts>-startroutine IS NOT INITIAL.
        ls_lookup_temp-codeid = <s_rsts>-startroutine.
        ls_lookup_temp-routinetype = 'RT10'.
        APPEND ls_lookup_temp TO lt_lookup_temp.
      ENDIF.
    ENDLOOP.

    SORT lt_lookup_temp BY codeid.
    LOOP AT lt_rsaabap INTO ls_rsaabap.
      CLEAR ls_lookup.
      READ TABLE lt_lookup_temp INTO ls_lookup
      WITH KEY codeid = ls_rsaabap-codeid BINARY SEARCH.
      IF sy-subrc = 0.
        CLEAR ls_result.
        IF ls_lookup-iobjnm IS NOT INITIAL.
          ls_lookup-routinetype = 'RT11'.
        ENDIF.
        MOVE-CORRESPONDING ls_lookup  TO ls_result.
        MOVE-CORRESPONDING ls_rsaabap TO ls_result.

        APPEND ls_result TO gt_result.
      ENDIF.
    ENDLOOP.

  ENDIF.
ENDIF.

"---------------------------------------------------------------------
" Scan routines of Transformations (7.x)
"---------------------------------------------------------------------
IF i_trans = 'X'.

  CLEAR lt_rsaabap.
  LOOP AT gt_rsaabap INTO ls_rsaabap WHERE codetp = 'TF'.
    APPEND ls_rsaabap TO lt_rsaabap.
  ENDLOOP.
  IF NOT lt_rsaabap IS INITIAL.

    "------- Transformations (rules)
    CLEAR lt_lookup_temp.
    SELECT targetname targettype targetsubtype sourcename sourcetype
       sourcesubtype codeid a~tranid fieldnm
      INTO TABLE lt_lookup_temp
      FROM rstran AS a
      INNER JOIN rstranroutmap AS b ON a~tranid = b~tranid
      FOR ALL ENTRIES IN lt_rsaabap
      WHERE
        expert    = ''                AND
        b~codeid  = lt_rsaabap-codeid AND
        a~objvers = 'A'               AND
        b~objvers = 'A'               AND
        param_id  = 1                 AND
        paramtype = 1.                                    "#EC *

    LOOP AT lt_lookup_temp INTO ls_lookup.
      ls_lookup-routinetype = 'RT04'.
      APPEND ls_lookup TO lt_lookup_finder.
    ENDLOOP.

    "------- Transformations (start routine)
    CLEAR lt_lookup_temp.
    SELECT targetname targettype targetsubtype sourcename sourcetype
       sourcesubtype startroutine tranid
       INTO TABLE lt_lookup_temp FROM rstran
      FOR ALL ENTRIES IN lt_rsaabap
      WHERE
        startroutine = lt_rsaabap-codeid AND
        objvers      = 'A'.                               "#EC *

    LOOP AT lt_lookup_temp INTO ls_lookup.
      ls_lookup-routinetype = 'RT01'.
      APPEND ls_lookup TO lt_lookup_finder.
    ENDLOOP.

    "------- Transformations (end routine)
    CLEAR lt_lookup_temp.
    SELECT targetname targettype targetsubtype sourcename sourcetype
```

```
  sourcesubtype endroutine tranid
   INTO TABLE lt_lookup_temp FROM rstran
  FOR ALL ENTRIES IN lt_rsaabap
  WHERE
    endroutine = lt_rsaabap-codeid AND
    objvers    = 'A'.                                    "#EC *

LOOP AT lt_lookup_temp INTO ls_lookup.
  ls_lookup-routinetype = 'RT02'.
  APPEND ls_lookup TO lt_lookup_finder.
ENDLOOP.

"------- Transformations (expert routine)
CLEAR lt_lookup_temp.
SELECT targetname targettype targetsubtype sourcename sourcetype
   sourcesubtype expert tranid
   INTO TABLE lt_lookup_temp FROM rstran
  FOR ALL ENTRIES IN lt_rsaabap
  WHERE
    expert  = lt_rsaabap-codeid AND
    objvers = 'A'.                                       "#EC *

LOOP AT lt_lookup_temp INTO ls_lookup.
  ls_lookup-routinetype = 'RT03'.
  APPEND ls_lookup TO lt_lookup_finder.
ENDLOOP.

"------- Transformations (Global declaration part 1)
CLEAR lt_lookup_temp.
SELECT targetname targettype targetsubtype sourcename sourcetype
   sourcesubtype glbcode tranid
   INTO TABLE lt_lookup_temp FROM rstran
  FOR ALL ENTRIES IN lt_rsaabap
  WHERE
    glbcode = lt_rsaabap-codeid AND
    objvers = 'A'.                                       "#EC *

LOOP AT lt_lookup_temp INTO ls_lookup.
  ls_lookup-routinetype = 'RT05'.
  APPEND ls_lookup TO lt_lookup_finder.
ENDLOOP.

"------- Transformations (Global declaration part 2)
CLEAR lt_lookup_temp.
SELECT targetname targettype targetsubtype sourcename sourcetype
   sourcesubtype glbcode2 tranid
   INTO TABLE lt_lookup_temp FROM rstran
  FOR ALL ENTRIES IN lt_rsaabap
  WHERE
    glbcode2 = lt_rsaabap-codeid AND
    objvers  = 'A'.                                      "#EC *

LOOP AT lt_lookup_temp INTO ls_lookup.
  ls_lookup-routinetype = 'RT06'.
  APPEND ls_lookup TO lt_lookup_finder.
ENDLOOP.
"-------
SORT lt_lookup_finder BY codeid.
LOOP AT lt_rsaabap INTO ls_rsaabap.
  CLEAR ls_lookup.
  READ TABLE lt_lookup_finder INTO ls_lookup
   WITH KEY codeid = ls_rsaabap-codeid BINARY SEARCH.
  IF sy-subrc = 0.
    CLEAR ls_result.
    IF ls_lookup-sourcetype = 'RSDS'.
      CONDENSE ls_lookup-sourcename.
    ENDIF.
    MOVE-CORRESPONDING ls_lookup  TO ls_result.
    MOVE-CORRESPONDING ls_rsaabap TO ls_result.
    "Get description of Transformation
    CLEAR lv_txtlg.
    SELECT SINGLE txtlg INTO lv_txtlg FROM rstrant
      WHERE langu = sy-langu
       AND  tranid = ls_lookup-tranid
       AND objvers = 'A'.
    IF sy-subrc <> 0.
      "If no text found derive description of the Transformation
      CONCATENATE ls_lookup-sourcetype ls_lookup-sourcename '->'
```

```
                    ls_lookup-targettype ls_lookup-targetname
                     INTO lv_txtlg SEPARATED BY space.
      ENDIF.
      ls_result-txtlg = lv_txtlg.

      APPEND ls_result TO gt_result.
    ENDIF.
  ENDLOOP.
ENDIF.

"Start Change ADü_20180618 - DP-1210
"handle AMDP Scripts
"RT20 -> HANA Script (Expert Routine)
"RT21 -> AMDP Start Routine
"RT22 -> AMDP End Routine
"RT23 -> AMDP Expert Routine
"RT24 -> AMDP InfoObject Routine

"HANA Script
IF lt_g_script IS NOT INITIAL.
  CLEAR lt_lookup_finder.
  "------- Transformations - HANA Script (Expert Routine)
  CLEAR lt_lookup_temp.
  SELECT targetname targettype targetsubtype sourcename sourcetype
     sourcesubtype tranid
    INTO CORRESPONDING FIELDS OF TABLE lt_lookup_temp
    FROM rstran
    FOR ALL ENTRIES IN lt_g_script
    WHERE tranid  = lt_g_script-tranid
    AND   objvers = 'A'.                                "#EC *

  LOOP AT lt_lookup_temp INTO ls_lookup.
    ls_lookup-routinetype = 'RT20'.
    APPEND ls_lookup TO lt_lookup_finder.
  ENDLOOP.
  "-------
  SORT lt_lookup_finder BY codeid.
  LOOP AT lt_g_script INTO ls_g_script.
    CLEAR ls_lookup.
    READ TABLE lt_lookup_finder INTO ls_lookup
     WITH KEY tranid = ls_g_script-tranid BINARY SEARCH.
    IF sy-subrc = 0.
      CLEAR ls_result.
      IF ls_lookup-sourcetype = 'RSDS'.
        CONDENSE ls_lookup-sourcename.
      ENDIF.
      MOVE-CORRESPONDING ls_lookup  TO ls_result.
      MOVE-CORRESPONDING ls_g_script TO ls_result.
      ls_result-line = ls_g_script-script.
      "Get description of Transformation
      CLEAR lv_txtlg.
      SELECT SINGLE txtlg INTO lv_txtlg FROM rstrant
        WHERE langu = sy-langu
         AND  tranid = ls_lookup-tranid
         AND objvers = 'A'.
      IF sy-subrc <> 0.
        "If no text found derive description of the Transformation
        CONCATENATE ls_lookup-sourcetype ls_lookup-sourcename '->'
                    ls_lookup-targettype ls_lookup-targetname
                     INTO lv_txtlg SEPARATED BY space.
      ENDIF.
      ls_result-txtlg = lv_txtlg.
      ls_result-codetp = 'TF'.

      APPEND ls_result TO gt_result.
    ENDIF.
  ENDLOOP.

ENDIF.

"AMDP
IF lt_g_sscript IS NOT INITIAL.
  CLEAR lt_lookup_finder.
  "------- Transformations - AMDP Start Routine
  CLEAR lt_lookup_temp.
  SELECT targetname targettype targetsubtype sourcename sourcetype
     sourcesubtype startroutine tranid
     INTO TABLE lt_lookup_temp FROM rstran
```

```
  FOR ALL ENTRIES IN lt_g_sscript
  WHERE
    startroutine = lt_g_sscript-codeid AND
    objvers      = 'A'.                                  "#EC *

LOOP AT lt_lookup_temp INTO ls_lookup.
  ls_lookup-routinetype = 'RT21'.
  APPEND ls_lookup TO lt_lookup_finder.
ENDLOOP.

"------- Transformations - AMDP End Routine
CLEAR lt_lookup_temp.
SELECT targetname targettype targetsubtype sourcename sourcetype
  sourcesubtype endroutine tranid
   INTO TABLE lt_lookup_temp FROM rstran
  FOR ALL ENTRIES IN lt_g_sscript
  WHERE
    endroutine = lt_g_sscript-codeid AND
    objvers    = 'A'.                                   "#EC *

LOOP AT lt_lookup_temp INTO ls_lookup.
  ls_lookup-routinetype = 'RT22'.
  APPEND ls_lookup TO lt_lookup_finder.
ENDLOOP.

"------- Transformations - AMDP Expert Routine
CLEAR lt_lookup_temp.
SELECT targetname targettype targetsubtype sourcename sourcetype
   sourcesubtype expert tranid
   INTO TABLE lt_lookup_temp FROM rstran
  FOR ALL ENTRIES IN lt_g_sscript
  WHERE
    expert  = lt_g_sscript-codeid AND
    objvers = 'A'.                                      "#EC *

LOOP AT lt_lookup_temp INTO ls_lookup.
  ls_lookup-routinetype = 'RT23'.
  APPEND ls_lookup TO lt_lookup_finder.
ENDLOOP.

"------- Transformations - AMDP InfoObject Routine
CLEAR lt_lookup_temp.
SELECT targetname targettype targetsubtype sourcename sourcetype
   sourcesubtype codeid a~tranid fieldnm
  INTO TABLE lt_lookup_temp
  FROM rstran AS a
  INNER JOIN rstranroutmap AS b ON a~tranid = b~tranid
  FOR ALL ENTRIES IN lt_g_sscript
  WHERE
    expert    = ''                  AND
    b~codeid  = lt_g_sscript-codeid AND
    a~objvers = 'A'                 AND
    b~objvers = 'A'                 AND
    param_id  = 1                   AND
    paramtype = 1.                                      "#EC *

LOOP AT lt_lookup_temp INTO ls_lookup.
  ls_lookup-routinetype = 'RT24'.
  APPEND ls_lookup TO lt_lookup_finder.
ENDLOOP.

"-------
SORT lt_lookup_finder BY codeid.
LOOP AT lt_g_sscript INTO ls_g_sscript.
  CLEAR ls_lookup.
  READ TABLE lt_lookup_finder INTO ls_lookup
   WITH KEY codeid = ls_g_sscript-codeid BINARY SEARCH.
  IF sy-subrc = 0.
    CLEAR ls_result.
    IF ls_lookup-sourcetype = 'RSDS'.
      CONDENSE ls_lookup-sourcename.
    ENDIF.
    MOVE-CORRESPONDING ls_lookup  TO ls_result.
    MOVE-CORRESPONDING ls_g_sscript TO ls_result.
    ls_result-line = ls_g_sscript-script.
    "Get description of Transformation
    CLEAR lv_txtlg.
    SELECT SINGLE txtlg INTO lv_txtlg FROM rstrant
```

```
            WHERE langu = sy-langu
             AND  tranid = ls_lookup-tranid
             AND objvers = 'A'.
          IF sy-subrc <> 0.
            "If no text found derive description of the Transformation
            CONCATENATE ls_lookup-sourcetype ls_lookup-sourcename '->'
                        ls_lookup-targettype ls_lookup-targetname
                         INTO lv_txtlg SEPARATED BY space.
          ENDIF.
          ls_result-txtlg = lv_txtlg.
          ls_result-codetp = 'TF'.

          APPEND ls_result TO gt_result.
        ENDIF.
      ENDLOOP.
    ENDIF.
    "End Change ADü_20180618 - DP-1210

    "BW4/HANA
    IF lt_g_rstransteprout IS NOT INITIAL.
      CLEAR lt_lookup_finder.
      CLEAR lt_lookup_temp.
      "------- Transformations - Start Routine
      SELECT ('targetname targettype targetsubtype sourcename sourcetype sourcesubtype
startroutine tranid set_runtime')
         INTO TABLE lt_lookup_temp FROM rstran
        FOR ALL ENTRIES IN lt_g_rstransteprout
        WHERE
          startroutine = lt_g_rstransteprout-codeid AND
          objvers      = 'A'.                                  "#EC *

      LOOP AT lt_lookup_temp INTO ls_lookup.
        IF ls_lookup-set_runtime = 'O'.
          ls_lookup-routinetype = 'RT34'.
        ELSE.
          ls_lookup-routinetype = 'RT30'.
        ENDIF.
        APPEND ls_lookup TO lt_lookup_finder.
      ENDLOOP.
      "------- Transformations - End Routine
      CLEAR lt_lookup_temp.
      SELECT ('targetname targettype targetsubtype sourcename sourcetype sourcesubtype endroutine
tranid set_runtime')
         INTO TABLE lt_lookup_temp FROM rstran
        FOR ALL ENTRIES IN lt_g_rstransteprout
        WHERE
          endroutine = lt_g_rstransteprout-codeid AND
          objvers    = 'A'.                                   "#EC *

      LOOP AT lt_lookup_temp INTO ls_lookup.
        IF ls_lookup-set_runtime = 'O'.
          ls_lookup-routinetype = 'RT35'.
        ELSE.
          ls_lookup-routinetype = 'RT31'.
        ENDIF.
        APPEND ls_lookup TO lt_lookup_finder.
      ENDLOOP.
      "------- Transformations - Expert Routine
      CLEAR lt_lookup_temp.
      SELECT ('targetname targettype targetsubtype sourcename sourcetype sourcesubtype expert
tranid set_runtime')
         INTO TABLE lt_lookup_temp FROM rstran
        FOR ALL ENTRIES IN lt_g_rstransteprout
        WHERE
          expert  = lt_g_rstransteprout-codeid AND
          objvers = 'A'.                                      "#EC *

      LOOP AT lt_lookup_temp INTO ls_lookup.
        IF ls_lookup-set_runtime = 'O'.
          ls_lookup-routinetype = 'RT37'.
        ELSE.
          ls_lookup-routinetype = 'RT33'.
        ENDIF.
        APPEND ls_lookup TO lt_lookup_finder.
      ENDLOOP.
      "------- Transformations - InfoObject Routine
      CLEAR lt_lookup_temp.
```

```
      SELECT ('targetname targettype targetsubtype sourcename sourcetype sourcesubtype codeid
a~tranid a~set_runtime fieldnm')
        INTO TABLE lt_lookup_temp
        FROM rstran AS a
        INNER JOIN rstranroutmap AS b ON a~tranid = b~tranid
        FOR ALL ENTRIES IN lt_g_rstransteprout
        WHERE
          expert    = ''                  AND
          b~codeid  = lt_g_rstransteprout-codeid AND
          a~objvers = 'A'                 AND
          b~objvers = 'A'                 AND
          param_id  = 1                   AND
          paramtype = 1.                                    "#EC *

      LOOP AT lt_lookup_temp INTO ls_lookup.
        IF ls_lookup-set_runtime = 'O'.
          ls_lookup-routinetype = 'RT36'.
        ELSE.
          ls_lookup-routinetype = 'RT32'.
        ENDIF.
        APPEND ls_lookup TO lt_lookup_finder.
      ENDLOOP.
      "------- Transformations (Global declaration part 1)
      CLEAR lt_lookup_temp.
      SELECT targetname targettype targetsubtype sourcename sourcetype
         sourcesubtype glbcode tranid
         INTO TABLE lt_lookup_temp FROM rstran
        FOR ALL ENTRIES IN lt_g_rstransteprout
        WHERE
          glbcode = lt_g_rstransteprout-codeid AND
          objvers = 'A'.                                    "#EC *

      LOOP AT lt_lookup_temp INTO ls_lookup.
        ls_lookup-routinetype = 'RT05'.
        APPEND ls_lookup TO lt_lookup_finder.
      ENDLOOP.
      "------- Transformations (Global declaration part 2)
      CLEAR lt_lookup_temp.
      SELECT targetname targettype targetsubtype sourcename sourcetype
         sourcesubtype glbcode2 tranid
         INTO TABLE lt_lookup_temp FROM rstran
        FOR ALL ENTRIES IN lt_g_rstransteprout
        WHERE
          glbcode2 = lt_g_rstransteprout-codeid AND
          objvers  = 'A'.                                   "#EC *

      LOOP AT lt_lookup_temp INTO ls_lookup.
        ls_lookup-routinetype = 'RT06'.
        APPEND ls_lookup TO lt_lookup_finder.
      ENDLOOP.
      "-------
      SORT lt_lookup_finder BY codeid.
      LOOP AT lt_g_rstransteprout INTO ls_g_rstransteprout.
        CLEAR ls_lookup.
        READ TABLE lt_lookup_finder INTO ls_lookup
         WITH KEY codeid = ls_g_rstransteprout-codeid BINARY SEARCH.
        IF sy-subrc = 0.
          CLEAR ls_result.
          IF ls_lookup-sourcetype = 'RSDS'.
            CONDENSE ls_lookup-sourcename.
          ENDIF.
          MOVE-CORRESPONDING ls_lookup  TO ls_result.
          MOVE-CORRESPONDING ls_g_rstransteprout TO ls_result.
          ls_result-line = ls_g_rstransteprout-code.
          "Get description of Transformation
          CLEAR lv_txtlg.
          SELECT SINGLE txtlg INTO lv_txtlg FROM rstrant
            WHERE langu = sy-langu
             AND  tranid = ls_lookup-tranid
             AND objvers = 'A'.
          IF sy-subrc <> 0.
            "If no text found derive description of the Transformation
            CONCATENATE ls_lookup-sourcetype ls_lookup-sourcename '->'
                        ls_lookup-targettype ls_lookup-targetname
                         INTO lv_txtlg SEPARATED BY space.
          ENDIF.
          ls_result-txtlg = lv_txtlg.
          ls_result-codetp = 'TF'.
```

```
          APPEND ls_result TO gt_result.
        ENDIF.
      ENDLOOP.
    ENDIF.

  ENDIF.

  "---------------------------------------------------------------------
  " Scan InfoObject (Char.) Transfer-Routines
  "---------------------------------------------------------------------
  IF i_iobjrout = 'X'.

    TYPES:
      BEGIN OF s_chabas,
        chabasnm LIKE rsdchabas-chabasnm,
        iobjrout LIKE rsdchabas-iobjrout,
      END OF s_chabas,
      t_chabas TYPE STANDARD TABLE OF s_chabas.

    DATA lt_chabas TYPE t_chabas.
    FIELD-SYMBOLS <s_chabas> TYPE s_chabas.

    CLEAR lt_rsaabap.
    LOOP AT gt_rsaabap INTO ls_rsaabap WHERE codetp = 'IC'.
      APPEND ls_rsaabap TO lt_rsaabap.
    ENDLOOP.
    IF NOT lt_rsaabap IS INITIAL.

      CLEAR: lt_lookup_temp, lt_chabas.
      SELECT  chabasnm iobjrout
        INTO TABLE lt_chabas
        FROM rsdchabas
        FOR ALL ENTRIES IN lt_rsaabap
        WHERE
          iobjrout = lt_rsaabap-codeid AND
          objvers    = 'A'.                                    "#EC *

      LOOP AT lt_chabas ASSIGNING <s_chabas>.
        CLEAR ls_lookup.
        ls_lookup-iobjnm = <s_chabas>-chabasnm.
        ls_lookup-codeid = <s_chabas>-iobjrout.
        APPEND ls_lookup TO lt_lookup_temp.
      ENDLOOP.

      SORT lt_lookup_temp BY codeid.
      LOOP AT lt_rsaabap INTO ls_rsaabap.
        CLEAR ls_lookup.
        READ TABLE lt_lookup_temp INTO ls_lookup
         WITH KEY codeid = ls_rsaabap-codeid BINARY SEARCH.
        IF sy-subrc = 0.
          CLEAR ls_result.
          ls_lookup-routinetype = 'RT15'.
          MOVE-CORRESPONDING ls_lookup  TO ls_result.
          MOVE-CORRESPONDING ls_rsaabap TO ls_result.
          APPEND ls_result TO gt_result.
        ENDIF.
      ENDLOOP.

    ENDIF.
  ENDIF.

ENDFORM.                    "scan_rsaabap


*&---------------------------------------------------------------------*
*&      Form  scan_dtp
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM scan_dtp
  TABLES lt_string t_devcl_range
    USING i_skip_comment i_case i_regex.

  DATA:
    l_t_dtp_range     TYPE t_devcl_range,
    l_s_dtp_range     TYPE s_devcl_range,
```

```
  lt_obj            TYPE STANDARD TABLE OF tadir-obj_name,
  ls_match_result   TYPE match_result,
  l_dtpnm           TYPE rsbkdtpnm,
  ls_rsbkdtpt       LIKE rsbkdtpt,
  lt_rsbkdtpt       LIKE TABLE OF rsbkdtpt,
  lr_cl_rsbk_dtp    TYPE REF TO cl_rsbk_dtp,
  lr_cl_rsbc_filter TYPE REF TO cl_rsbc_filter,
  ls_dtprule        TYPE mch_s_sourcecode,
  lt_dtprule        TYPE mch_t_sourcecode,
  ls_result         TYPE s_result.

FIELD-SYMBOLS:
  <string>  TYPE s_string,
  <s_range> TYPE rsrange,
  <s_obj>   LIKE LINE OF lt_obj.
"---------------------------------------------------------------------

SELECT obj_name INTO TABLE lt_obj FROM tadir
  WHERE
    pgmid    = 'R3TR'           AND
    object   = 'DTPA'.                                    "#EC *

SELECT * INTO TABLE lt_rsbkdtpt FROM rsbkdtpt
  WHERE
    langu   = sy-langu AND
    objvers = 'A'.

SORT lt_rsbkdtpt BY dtp.

LOOP AT lt_obj ASSIGNING <s_obj>.

  CLEAR:
    lr_cl_rsbk_dtp,
    lr_cl_rsbc_filter.

  "===== Get Filter-Routines =====
  l_dtpnm = <s_obj>.
  lr_cl_rsbk_dtp    = cl_rsbk_dtp=>factory( l_dtpnm ).
  TRY.
      lr_cl_rsbc_filter = lr_cl_rsbk_dtp->get_obj_ref_filter( ).
    CATCH cx_rs_access_error.
      CONTINUE.
  ENDTRY.

  LOOP AT lr_cl_rsbc_filter->n_t_dtprule INTO ls_dtprule.

    LOOP AT lt_string ASSIGNING <string>.
      CLEAR ls_match_result.
      IF i_case = ' ' AND i_regex = ' '.
        FIND FIRST OCCURRENCE OF <string> IN ls_dtprule-line
          IN CHARACTER MODE
          IGNORING CASE
          RESULTS ls_match_result.
      ELSEIF i_case = 'X' AND i_regex = ' '.
        FIND FIRST OCCURRENCE OF <string> IN ls_dtprule-line
          IN CHARACTER MODE
          RESPECTING CASE
          RESULTS ls_match_result.
      ELSEIF i_case = ' ' AND i_regex = 'X'.
        FIND FIRST OCCURRENCE OF REGEX <string> IN ls_dtprule-line
          IN CHARACTER MODE
          IGNORING CASE
          RESULTS ls_match_result.
      ELSEIF i_case = 'X' AND i_regex = 'X'.
        FIND FIRST OCCURRENCE OF REGEX <string> IN ls_dtprule-line
          IN CHARACTER MODE
          RESPECTING CASE
          RESULTS ls_match_result.
      ENDIF.
      IF ls_match_result-offset <> 0.
        CLEAR ls_result.
        ls_result-codetp      = 'DTP'.
        ls_result-targetname  = <s_obj>.
        ls_result-routinetype = 'RT13'.
        ls_result-iobjnm      = ls_dtprule-field.
        ls_result-line_no     = ls_dtprule-line_no.
        ls_result-line        = ls_dtprule-line.
        "Get description
```

```
          READ TABLE lt_rsbkdtpt INTO ls_rsbkdtpt
           WITH KEY dtp = <s_obj> BINARY SEARCH.
          IF sy-subrc = 0.
            ls_result-txtlg = ls_rsbkdtpt-txtlg.
          ENDIF.
          APPEND ls_result TO gt_result.
        ENDIF.
      ENDLOOP.

    ENDLOOP.

  ENDLOOP.

ENDFORM.                    "scan_dtp



*&---------------------------------------------------------------------*
*&      Form  scan_ip
*&---------------------------------------------------------------------*
FORM scan_ip
    TABLES lt_string t_devcl_range
    USING i_skip_comment i_case i_regex.
  DATA:
    l_t_devcl_range TYPE t_devcl_range,
    l_s_devcl_range TYPE s_devcl_range,
    lt_obj          TYPE STANDARD TABLE OF tadir-obj_name,
    ls_rsldprule    LIKE rsldprule,
    lt_rsldprule    LIKE TABLE OF ls_rsldprule,
    ls_match_result TYPE match_result,
    ls_rsldpiot     LIKE rsldpiot,
    lt_rsldpiot     LIKE TABLE OF rsldpiot,
    ls_result       TYPE s_result.

  FIELD-SYMBOLS:
    <string>  TYPE s_string,
    <s_range> TYPE rsrange,
    <s_obj>   LIKE LINE OF lt_obj.


  LOOP AT t_devcl_range ASSIGNING <s_range>.
    l_s_devcl_range-sign   = <s_range>-sign.
    l_s_devcl_range-option = <s_range>-option.
    l_s_devcl_range-low    = <s_range>-low.
    l_s_devcl_range-high   = <s_range>-high.
    APPEND l_s_devcl_range TO l_t_devcl_range.
  ENDLOOP.

  "---------------------------------------------------------------------

  SELECT obj_name INTO TABLE lt_obj FROM tadir
    WHERE
      pgmid    = 'R3TR'           AND
      object   = 'ISIP'.                                    "#EC *

  SELECT * INTO TABLE lt_rsldpiot FROM rsldpiot
    WHERE
      langu = sy-langu AND
      objvers = 'A'.

  SORT lt_rsldpiot BY logdpid.

  LOOP AT lt_obj ASSIGNING <s_obj>.

    CLEAR lt_rsldprule.
    SELECT
        r~logdpid
        r~objvers
        r~fieldname
        r~lnr
        r~iobjnm
        line
      INTO TABLE lt_rsldprule
      FROM rsldpsel AS s INNER JOIN rsldprule AS r
      ON s~logdpid = r~logdpid AND
         s~objvers = r~objvers AND
         s~fieldname = r~fieldname
      WHERE
```

```
        s~logdpid = <s_obj> AND
        s~objvers = 'A'.

    LOOP AT lt_rsldprule INTO ls_rsldprule.

      LOOP AT lt_string ASSIGNING <string>.

        CLEAR ls_match_result.
        IF i_case = ' ' AND i_regex = ' '.
          FIND FIRST OCCURRENCE OF <string> IN ls_rsldprule-line
            IN CHARACTER MODE
            IGNORING CASE
            RESULTS ls_match_result.
        ELSEIF i_case = 'X' AND i_regex = ' '.
          FIND FIRST OCCURRENCE OF <string> IN ls_rsldprule-line
            IN CHARACTER MODE
            RESPECTING CASE
            RESULTS ls_match_result.
        ELSEIF i_case = ' ' AND i_regex = 'X'.
          FIND FIRST OCCURRENCE OF REGEX <string> IN ls_rsldprule-line
            IN CHARACTER MODE
            IGNORING CASE
            RESULTS ls_match_result.
        ELSEIF i_case = 'X' AND i_regex = 'X'.
          FIND FIRST OCCURRENCE OF REGEX <string> IN ls_rsldprule-line
            IN CHARACTER MODE
            RESPECTING CASE
            RESULTS ls_match_result.
        ENDIF.

        IF ls_match_result-offset <> 0.
          CLEAR ls_result.
          ls_result-codetp      = 'IP'.
          ls_result-targetname  = <s_obj>.
          ls_result-routinetype = 'RT14'.
          ls_result-txtlg       = ls_rsldprule-fieldname.
          ls_result-iobjnm      = ls_rsldprule-iobjnm.
          ls_result-line_no     = sy-tabix.
          ls_result-line        = ls_rsldprule-line.
          READ TABLE lt_rsldpiot INTO ls_rsldpiot
          WITH KEY logdpid = ls_rsldprule-logdpid BINARY SEARCH.
          IF sy-subrc = 0.
            ls_result-txtlg = ls_rsldpiot-text.
          ENDIF.
          APPEND ls_result TO gt_result.
        ENDIF.

      ENDLOOP.

    ENDLOOP.

  ENDLOOP.
ENDFORM.                    "scan_ip


*&---------------------------------------------------------------------*
*&      Form  Get_REPS
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM get_reps
  TABLES t_devcl_range t_reps_range t_subc_range.

  DATA:
    l_t_devcl_range TYPE t_devcl_range,
    l_s_devcl_range TYPE s_devcl_range,
    l_t_name_range  TYPE t_name_range,
    l_s_name_range  TYPE s_name_range,
    l_t_subc_range  TYPE t_subc_range,
    l_s_subc_range  TYPE s_subc_range.

  FIELD-SYMBOLS:
    <s_range>       TYPE rsrange.

  "---------------------------------------------------------------------
  " Range Importparam. in Selektionstab. mit passendem Typ übertragen
  "---------------------------------------------------------------------
  LOOP AT t_devcl_range ASSIGNING <s_range>.
```

```
    l_s_devcl_range-sign   = <s_range>-sign.
    l_s_devcl_range-option = <s_range>-option.
    l_s_devcl_range-low    = <s_range>-low.
    l_s_devcl_range-high   = <s_range>-high.
    APPEND l_s_devcl_range TO l_t_devcl_range.
  ENDLOOP.

  LOOP AT t_reps_range ASSIGNING <s_range>.
    l_s_name_range-sign   = <s_range>-sign.
    l_s_name_range-option = <s_range>-option.
    l_s_name_range-low    = <s_range>-low.
    l_s_name_range-high   = <s_range>-high.
    APPEND l_s_name_range TO l_t_name_range.
  ENDLOOP.

  LOOP AT t_subc_range ASSIGNING <s_range>.
    l_s_subc_range-sign   = <s_range>-sign.
    l_s_subc_range-option = <s_range>-option.
    l_s_subc_range-low    = <s_range>-low.
    l_s_subc_range-high   = <s_range>-high.
    APPEND l_s_subc_range TO l_t_subc_range.
  ENDLOOP.
  "---------------------------------------------------------------------

  SELECT object obj_name INTO TABLE gt_object FROM tadir
     AS a INNER JOIN trdir AS r ON a~obj_name = r~name
      WHERE
       pgmid    = 'R3TR'   AND
       object   = 'PROG'   AND
       obj_name IN l_t_name_range  AND
       devclass IN l_t_devcl_range AND
       subc     IN l_t_subc_range.                          "#EC *

ENDFORM.                    "Get_REPS

*&---------------------------------------------------------------------*
*&      Form  get_FUNC
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->I_T_DEVCL_RANGE  text
*      -->I_T_FUGR_RANGE   text
*----------------------------------------------------------------------*
FORM get_func
  TABLES t_devcl_range t_fugr_range.

  DATA:
    l_t_devcl_range TYPE t_devcl_range,
    l_s_devcl_range TYPE s_devcl_range,
    l_t_fugr_range  TYPE t_fugr_range,
    l_s_fugr_range  TYPE s_fugr_range,
    lt_obj          TYPE STANDARD TABLE OF s_object,
    l_fgroup        TYPE rs38l-area,
    l_program       TYPE progname.

  FIELD-SYMBOLS:
    <s_range> TYPE rsrange,
    <s_obj>   LIKE LINE OF lt_obj.

  "---------------------------------------------------------------------
  " Range Importparam. in Selektionstab. mit passendem Typ übertragen
  "---------------------------------------------------------------------
  LOOP AT t_devcl_range ASSIGNING <s_range>.
    l_s_devcl_range-sign   = <s_range>-sign.
    l_s_devcl_range-option = <s_range>-option.
    l_s_devcl_range-low    = <s_range>-low.
    l_s_devcl_range-high   = <s_range>-high.
    APPEND l_s_devcl_range TO l_t_devcl_range.
  ENDLOOP.

  LOOP AT t_fugr_range ASSIGNING <s_range>.
    l_s_fugr_range-sign   = <s_range>-sign.
    l_s_fugr_range-option = <s_range>-option.
    l_s_fugr_range-low    = <s_range>-low.
    l_s_fugr_range-high   = <s_range>-high.
    APPEND l_s_fugr_range TO l_t_fugr_range.
```

```
  ENDLOOP.
  "---------------------------------------------------------------------

  SELECT object obj_name INTO TABLE lt_obj FROM tadir
    WHERE
      pgmid    = 'R3TR'           AND
      object   = 'FUGR'           AND
      obj_name IN l_t_fugr_range  AND
      devclass IN l_t_devcl_range.                          "#EC *

  LOOP AT lt_obj ASSIGNING <s_obj>.
    l_fgroup = <s_obj>-obj_name.
    CLEAR l_program.

    CALL FUNCTION 'FUNCTION_INCLUDE_CONCATENATE'
      CHANGING
        program                 = l_program
        complete_area           = l_fgroup
      EXCEPTIONS
        not_enough_input        = 1
        no_function_pool        = 2
        delimiter_wrong_position = 3
        OTHERS                  = 4.

    CHECK sy-subrc IS INITIAL AND l_program IS NOT INITIAL.
    <s_obj>-object = 'FUGR'.
    <s_obj>-incl_name = l_program.
    APPEND <s_obj> TO gt_object.
  ENDLOOP.

ENDFORM.                    "get_FUNC


*&---------------------------------------------------------------------*
*&      Form  get_plse
*&---------------------------------------------------------------------*
*       Search in Planning Functions
*----------------------------------------------------------------------*
FORM get_plse
    TABLES lt_string t_devcl_range
    USING i_skip_comment i_case i_regex.

  DATA:
    lt_obj          TYPE STANDARD TABLE OF tadir-obj_name,
    ls_rsplf_srv_p  TYPE rsplf_srv_p,
    lt_rsplf_srv_p  TYPE TABLE OF rsplf_srv_p,
    ls_match_result TYPE match_result,
    ls_rsplf_srvt   TYPE rsplf_srvt,
    lt_rsplf_srvt   TYPE TABLE OF rsplf_srvt,
    ls_result       TYPE s_result.

  FIELD-SYMBOLS:
    <string>  TYPE s_string,
    <s_range> TYPE rsrange,
    <s_obj>   LIKE LINE OF lt_obj.
  "---------------------------------------------------------------------

  SELECT obj_name INTO TABLE lt_obj FROM tadir
    WHERE
      pgmid    = 'R3TR' AND
      object   = 'PLSE'.                                    "#EC *

  SELECT * INTO TABLE lt_rsplf_srvt FROM rsplf_srvt
    WHERE
      langu = sy-langu AND
      objvers = 'A'.                                        "#EC *

  SORT lt_rsplf_srvt BY srvnm.

  LOOP AT lt_obj ASSIGNING <s_obj>.

    CLEAR lt_rsplf_srv_p.
    SELECT * FROM rsplf_srv_p
      INTO TABLE lt_rsplf_srv_p
      WHERE
        srvnm = <s_obj> AND
        objvers = 'A'   AND
```

```
        parnm = 'FLINE'.                                    "#EC *


**new ADü_20170613
    DATA: lv_string   TYPE          string,
          lt_splitted TYPE TABLE OF string,
          ls_splitted TYPE          string,
          lv_index    TYPE          i VALUE 0.

    CLEAR: lv_string, lv_index.

    LOOP AT lt_rsplf_srv_p INTO ls_rsplf_srv_p.
      CONCATENATE lv_string ls_rsplf_srv_p-value INTO lv_string.
    ENDLOOP.

    SPLIT lv_string AT cl_abap_char_utilities=>cr_lf INTO TABLE lt_splitted.

    READ TABLE lt_rsplf_srv_p INTO ls_rsplf_srv_p INDEX 1.  "#EC *

    LOOP AT lt_splitted INTO ls_splitted.
      lv_index = lv_index + 1.

      "LOOP AT lt_rsplf_srv_p INTO ls_rsplf_srv_p.

**end new ADü_20170613

      LOOP AT lt_string ASSIGNING <string>.

        CLEAR ls_match_result.
        IF i_case = ' ' AND i_regex = ' '.
          FIND FIRST OCCURRENCE OF <string> IN ls_splitted "ls_rsplf_srv_p-value
            IN CHARACTER MODE
            IGNORING CASE
            RESULTS ls_match_result.
        ELSEIF i_case = 'X' AND i_regex = ' '.
          FIND FIRST OCCURRENCE OF <string> IN ls_splitted "ls_rsplf_srv_p-value
            IN CHARACTER MODE
            RESPECTING CASE
            RESULTS ls_match_result.
        ELSEIF i_case = ' ' AND i_regex = 'X'.
          FIND FIRST OCCURRENCE OF REGEX <string>
            IN ls_splitted "ls_rsplf_srv_p-value
            IN CHARACTER MODE
            IGNORING CASE
            RESULTS ls_match_result.
        ELSEIF i_case = 'X' AND i_regex = 'X'.
          FIND FIRST OCCURRENCE OF REGEX <string>
            IN ls_splitted "ls_rsplf_srv_p-value
            IN CHARACTER MODE
            RESPECTING CASE
            RESULTS ls_match_result.
        ENDIF.

        IF ls_match_result-offset <> 0.
          CLEAR ls_result.
          ls_result-codetp      = 'PF'.
          ls_result-targetname  = <s_obj>.
          ls_result-routinetype = 'RT19'.
          ls_result-txtlg       = ''.
          ls_result-iobjnm      = ''.
          ls_result-line_no     = lv_index. "ls_rsplf_srv_p-indx.
          ls_result-line        = ls_splitted. "ls_rsplf_srv_p-value.
          READ TABLE lt_rsplf_srvt INTO ls_rsplf_srvt
          WITH KEY srvnm = ls_rsplf_srv_p-srvnm BINARY SEARCH.
          IF sy-subrc = 0.
            ls_result-txtlg = ls_rsplf_srvt-txtlg.
          ENDIF.
          APPEND ls_result TO gt_result.
        ENDIF.

      ENDLOOP.

    ENDLOOP.

  ENDLOOP.

ENDFORM.                    "get_plse
```

```
*&---------------------------------------------------------------------*
*&      Form  get_clas
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM get_clas
  TABLES t_devcl_range t_clas_range.

  DATA:
    l_t_devcl_range TYPE t_devcl_range,
    l_s_devcl_range TYPE s_devcl_range,
    l_t_clas_range  TYPE t_clas_range,
    l_s_clas_range  TYPE s_clas_range,
    lt_obj          TYPE STANDARD TABLE OF s_object,
    l_clskey        TYPE seoclskey,
    l_obj           TYPE tadir-obj_name,
    lt_includes     TYPE seop_methods_w_include.

  FIELD-SYMBOLS:
    <s_range>   TYPE rsrange,
    <s_obj>     LIKE LINE OF lt_obj,
    <s_include> LIKE LINE OF lt_includes.

  "---------------------------------------------------------------------
  " Range Importparam. in Selektionstab. mit passendem Typ übertragen
  "---------------------------------------------------------------------
  LOOP AT t_devcl_range ASSIGNING <s_range>.
    l_s_devcl_range-sign   = <s_range>-sign.
    l_s_devcl_range-option = <s_range>-option.
    l_s_devcl_range-low    = <s_range>-low.
    l_s_devcl_range-high   = <s_range>-high.
    APPEND l_s_devcl_range TO l_t_devcl_range.
  ENDLOOP.

  LOOP AT t_clas_range ASSIGNING <s_range>.
    l_s_clas_range-sign   = <s_range>-sign.
    l_s_clas_range-option = <s_range>-option.
    l_s_clas_range-low    = <s_range>-low.
    l_s_clas_range-high   = <s_range>-high.
    APPEND l_s_clas_range TO l_t_clas_range.
  ENDLOOP.
  "---------------------------------------------------------------------

  SELECT object obj_name INTO TABLE lt_obj FROM tadir
    WHERE
      pgmid    = 'R3TR'          AND
      object   = 'CLAS'          AND
      obj_name IN l_t_clas_range AND
      devclass IN l_t_devcl_range.                          "#EC *

  LOOP AT lt_obj ASSIGNING <s_obj>.
    l_clskey = <s_obj>-obj_name.
* ts 28.10.2014 - Start comment
*    CALL FUNCTION 'SEO_CLASS_GET_METHOD_INCLUDES'
*      EXPORTING
*        clskey                      = l_clskey
*      IMPORTING
*        includes                    = lt_includes
*      EXCEPTIONS
*        _internal_class_not_existing = 1
*        OTHERS                      = 2.
*
*    LOOP AT lt_includes ASSIGNING <s_include>.
*      <s_obj>-object = 'METH'.
*      <s_obj>-incl_name = <s_include>-incname.
*      <s_obj>-obj_name = l_clskey.
*      <s_obj>-meth_name = <s_include>-cpdkey+30.
*      APPEND <s_obj> TO gt_object.
*    ENDLOOP.
* ts 28.10.2014 - End comment

* ts 28.10.2014 - Start insert
    CLASS cl_oo_include_naming DEFINITION LOAD.
    DATA lo_clif_incl_naming  TYPE REF TO if_oo_clif_incl_naming.
    DATA lo_class_incl_naming TYPE REF TO if_oo_class_incl_naming.
    DATA l_t_method_w_include TYPE seop_methods_w_include.
```

```
    FIELD-SYMBOLS: <s_method_w_include> TYPE seop_method_w_include.

    CALL METHOD cl_oo_include_naming=>get_instance_by_cifkey
      EXPORTING
        cifkey         = l_clskey
      RECEIVING
        cifref         = lo_clif_incl_naming
      EXCEPTIONS
        no_objecttype  = 1
        internal_error = 2
        OTHERS         = 3.

    IF sy-subrc = 0.
      lo_class_incl_naming ?= lo_clif_incl_naming.
      l_t_method_w_include =
        lo_class_incl_naming->get_all_method_includes( ).
      LOOP AT l_t_method_w_include ASSIGNING <s_method_w_include>.
        <s_obj>-object = 'METH'.
        <s_obj>-incl_name = <s_method_w_include>-incname.
        <s_obj>-obj_name  = l_clskey.
        <s_obj>-meth_name = <s_method_w_include>-cpdkey+30.
        APPEND <s_obj> TO gt_object.
      ENDLOOP.
      "ADü - Start insert 20170621
      "Public Section
      <s_obj>-object    = 'METH'.
      <s_obj>-incl_name = lo_class_incl_naming->public_section.
      <s_obj>-obj_name  = l_clskey.
      <s_obj>-meth_name = 'PUBLIC SECTION'.
      APPEND <s_obj> TO gt_object.
      "Private Section
      <s_obj>-object    = 'METH'.
      <s_obj>-incl_name = lo_class_incl_naming->private_section.
      <s_obj>-obj_name  = l_clskey.
      <s_obj>-meth_name = 'PRIVATE SECTION'.
      APPEND <s_obj> TO gt_object.
      "Protected Section
      <s_obj>-object    = 'METH'.
      <s_obj>-incl_name = lo_class_incl_naming->protected_section.
      <s_obj>-obj_name  = l_clskey.
      <s_obj>-meth_name = 'PROTECTED SECTION'.
      APPEND <s_obj> TO gt_object.
      "Local Implementations
      <s_obj>-object    = 'METH'.
      <s_obj>-incl_name = lo_class_incl_naming->locals_imp.
      <s_obj>-obj_name  = l_clskey.
      <s_obj>-meth_name = 'LOCAL IMPLEMENTATIONS'.
      APPEND <s_obj> TO gt_object.
      "ADü - End insert 20170621
    ENDIF.
* ts 28.10.2014 - End insert

  ENDLOOP.

ENDFORM.                    "get_clas


*&---------------------------------------------------------------------*
*&      Form  scan_include_code
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM scan_include_code
  TABLES lt_string
    USING i_skip_comment i_case i_regex.

  DATA:
    lt_object        TYPE STANDARD TABLE OF s_object,
    ls_object        TYPE s_object, "TS, 20140509
    lt_include       TYPE STANDARD TABLE OF tadir-obj_name,
    lv_program       TYPE sy-repid,
    lt_abap          TYPE abaptxt255_tab,
    lt_match_results TYPE match_result_tab,
    ls_match_result  LIKE LINE OF lt_match_results,
    ls_result        TYPE s_result,
    lv_func          TYPE rs38l-name,
    lv_func_incl     TYPE rs38l-name.
```

```
  FIELD-SYMBOLS:
    <string>  TYPE s_string,
    <s_range> TYPE rsrange,
    <obj>     TYPE s_object,
    <include> TYPE tadir-obj_name,
    <abap>    LIKE LINE OF lt_abap.


*---------------------------------------------------------------------
    "Get includes
    CLEAR lt_object.
    "for reports and classes we already have the includes
    LOOP AT gt_object ASSIGNING <obj> WHERE object = 'FUGR'.
      CLEAR lt_include.
      lv_program = <obj>-incl_name.

      CALL FUNCTION 'RS_GET_ALL_INCLUDES'
        EXPORTING
          program      = lv_program
        TABLES
          includetab   = lt_include
        EXCEPTIONS
          not_existent = 1
          no_program   = 2
          OTHERS       = 3.

      CHECK sy-subrc IS INITIAL.

      LOOP AT lt_include ASSIGNING <include>.
        ls_object-object = 'FUNC'.
        ls_object-incl_name = <include>.
        APPEND ls_object TO gt_object.
      ENDLOOP.

    ENDLOOP.

    SORT lt_object.
    DELETE ADJACENT DUPLICATES FROM lt_object.
    APPEND LINES OF lt_object TO gt_object.

    "Source scan
    LOOP AT gt_object ASSIGNING <obj>.

      IF <obj>-object = 'PROG'.
        <obj>-incl_name = <obj>-obj_name.
      ENDIF.
      READ REPORT <obj>-incl_name INTO lt_abap.
      IF sy-subrc IS NOT INITIAL.
        CONTINUE.
      ENDIF.

      LOOP AT lt_string ASSIGNING <string>.

        CLEAR lt_match_results.
        IF i_case = '' AND i_regex = ''.
          FIND ALL OCCURRENCES OF <string> IN TABLE lt_abap
            IN CHARACTER MODE
            IGNORING CASE
            RESULTS lt_match_results.
        ELSEIF i_case = 'X' AND i_regex = ''.
          FIND ALL OCCURRENCES OF <string> IN TABLE lt_abap
            IN CHARACTER MODE
            RESPECTING CASE
            RESULTS lt_match_results.
        ELSEIF i_case = '' AND i_regex = 'X'.
          FIND ALL OCCURRENCES OF REGEX <string> IN TABLE lt_abap
            IN CHARACTER MODE
            IGNORING CASE
            RESULTS lt_match_results.
        ELSEIF i_case = 'X' AND i_regex = 'X'.
          FIND ALL OCCURRENCES OF REGEX <string> IN TABLE lt_abap
            IN CHARACTER MODE
            RESPECTING CASE
            RESULTS lt_match_results.
        ENDIF.

        LOOP AT lt_match_results INTO ls_match_result.
```

```
        READ TABLE lt_abap
        INDEX ls_match_result-line ASSIGNING <abap>.
        IF sy-subrc = 0.
          CLEAR ls_result.
          ls_result-codetp = <obj>-object(2).
          "if FUNC: Get name of FM
          IF <obj>-object = 'FUNC' OR <obj>-object = 'FUGR'.
            CLEAR lv_func.

            CALL FUNCTION 'FUNCTION_INCLUDE_INFO'
            "IMPORTING
            "FUNCTAB        =
            "NAMESPACE      =
            "PNAME          =
              CHANGING
                funcname            = lv_func
                "GROUP          =
                include             = <obj>-incl_name
              EXCEPTIONS
                function_not_exists = 1
                include_not_exists  = 2
                group_not_exists    = 3
                no_selections       = 4
                no_function_include = 5
                OTHERS              = 6.
            IF sy-subrc = 0.
              <obj>-obj_name = lv_func.
            ENDIF.

          ENDIF.

          IF <obj>-object = 'METH'.
            ls_result-txtlg = <obj>-meth_name.
          ENDIF.
          ls_result-targetname = <obj>-obj_name.
          ls_result-sourcename = <obj>-incl_name.
          ls_result-line_no    = ls_match_result-line.
          ls_result-line       = <abap>.


          CASE ls_result-codetp.
            WHEN 'PR'.
              ls_result-routinetype = 'RT16'.
            WHEN 'ME'.
              ls_result-routinetype = 'RT17'.
            WHEN 'FU'.
              ls_result-routinetype = 'RT18'.
          ENDCASE.
          APPEND ls_result TO gt_result.
        ENDIF.
      ENDLOOP.

    ENDLOOP.

  ENDLOOP.

ENDFORM.                    "scan_include_code
```

6.6 Includes

7 Function Module Z_RFC_ENTITY_SYNC - RFC-Synchronisation of Entities

7.1 General Information

System	BI2
Function Module	Z_RFC_ENTITY_SYNC, RFC-Synchronisation of Entities
Function Pool	Z_DP
Remote	Yes
Last changed by	NMEYER
Last change (timestamp)	16/01/2018
Timestamp of documentation	11/08/2020 16:29:45

7.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
TIMESTMP	TYPE	RSTIMESTMP		X	X	
SYNC_WORKBENCH	TYPE	SONV-FLAG	'X'	X	X	
SYNC_REPORTING	TYPE	SONV-FLAG	'X'	X	X	
SYNC_CHAINS	TYPE	SONV-FLAG	'X'	X	X	
SYNC_PROG	TYPE	SONV-FLAG	'X'	X	X	
SYNC_FUNC	TYPE	SONV-FLAG	'X'	X	X	
SYNC_CLAS	TYPE	SONV-FLAG	'X'	X	X	
SYNC_IOBJ	TYPE	SONV-FLAG	'X'	X	X	
SYNC_IOBJCAT	TYPE	SONV-FLAG		X	X	
SYNC_AUTH	TYPE	SONV-FLAG	'X'	X	X	
SYNC_APD	TYPE	SONV-FLAG	'X'	X	X	
SYNC_BPC	TYPE	SONV-FLAG		X	X	
SYNC_WDYN	TYPE	SONV-FLAG	'X'	X	X	

7.3 Tables

Parameter		Associated Type	Opt.	Short text
T_DEVCLASS	LIKE	TCHLP		
T_LANGUAGE	LIKE	TCP0H		
T_RSDIOBJ	LIKE	RSDIOBJ		
T_RSDIOBJT	LIKE	RSDIOBJT		
T_RSDIOBCIOBJ	LIKE	RSDIOBCIOBJ		
T_RSDIOBCT	LIKE	RSDIOBCT		
T_RSBASIDOC	LIKE	RSBASIDOC		
T_RSDS	LIKE	RSDS		
T_RSDST	LIKE	RSDST		
T_RSOLTPSOURCE	LIKE	RSOLTPSOURCE		
T_RSOLTPSOURCET	LIKE	RSOLTPSOURCET		
T_RSTRANT	LIKE	RSTRANT		
T_RSIS	LIKE	RSIS		
T_RSIST	LIKE	RSIST		
T_RSKSNEW	LIKE	RSKSNEW		
T_RSKSNEWT	LIKE	RSKSNEWT		
T_RSTS	LIKE	RSTS		
T_RSDCUBET	LIKE	RSDCUBET		
T_RSDODSOT	LIKE	RSDODSOT		
T_RSQISET	LIKE	RSQISET		
T_RSQISETT	LIKE	RSQISETT		
T_RSBSPOKE	LIKE	RSBSPOKE		
T_RSBSPOKET	LIKE	RSBSPOKET		
T_RSBOHDEST	LIKE	RSBOHDEST		
T_RSBOHDESTT	LIKE	RSBOHDESTT		
T_RSPLS_ALVL	LIKE	RSPLS_ALVL		
T_RSPLS_ALVLT	LIKE	RSPLS_ALVLT		
T_RSPLF_SRV	LIKE	RSPLF_SRV		
T_RSPLF_SRVT	LIKE	RSPLF_SRVT		
T_RSPLS_SEQUENCE	LIKE	RSPLS_SEQUENCE		
T_RSPLS_SEQUENCET	LIKE	RSPLS_SEQUENCET		
T_RSPCCHAINATTR	LIKE	RSPCCHAINATTR		

T_RSPCCHAINT	LIKE	RSPCCHAINT		
T_V_REP_JOIN	LIKE	V_REP_JOIN		
T_V_REP_REUSE	LIKE	V_COMPDIR_COMPIC		
T_RSZGLOBV	LIKE	RSZGLOBV		
T_V_ELTDIR_TXT	LIKE	RSZELTTXT		
T_RSRWBINDEX	LIKE	RSRWBINDEX		
T_RSRWBINDEXT	LIKE	RSRWBINDEXT		
T_RSRWORKBOOK	LIKE	RSRWORKBOOK		
T_RSZWBTMPHEAD	LIKE	RSZWBTMPHEAD		
T_RSZWBTMPHEADTXT	LIKE	RSZWBTMPHEADTXT		
T_RSZWBTMPXREF	LIKE	RSZWBTMPXREF		
T_RSZWTEMPLATE	LIKE	RSZWTEMPLATE		
T_RSZWOBJTXT	LIKE	RSZWOBJTXT		
T_RSZWOBJXREF	LIKE	RSZWOBJXREF		
T_FUGR	LIKE	INFO_FUGRT		
T_FUNC	LIKE	INFO_FUNCT		
T_TADIR_PROG	LIKE	TRESN		
T_TRDIRT	LIKE	TRDIRT		
T_RSANT_PROCESST	LIKE	RSANT_PROCESST		
T_RSANT_PROCESS	LIKE	RSO_S_OBJECT_LIST		
T_RSDIOBC	LIKE	RSDIOBC		
T_RSISOSMAP	LIKE	RSISOSMAP		
T_RSUPDINFO	LIKE	RSUPDINFO		
T_RSDVUNI	LIKE	RSDVUNI		
T_AGR_DEFINE	LIKE	AGR_DEFINE		
T_AGR_TEXTS	LIKE	AGR_TEXTS		
T_RSDCUBEMULTI	LIKE	RSDCUBEMULTI		
T_RSQTOBJ	LIKE	RSQTOBJ		
T_RSDKYF	LIKE	RSDKYF		
T_VSEOCLASS	LIKE	VSEOCLASS		
T_WDYN	LIKE	TRESN		
T_WDY_COMPONENTT	LIKE	WDY_COMPONENTT		

7.4 Exceptions

Exception	Short text
RELEASE_1_9	

7.5 Source code Function module

```
FUNCTION Z_RFC_ENTITY_SYNC.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(TIMESTMP) TYPE  RSTIMESTMP OPTIONAL
*"     VALUE(SYNC_WORKBENCH) TYPE  SONV-FLAG DEFAULT 'X'
*"     VALUE(SYNC_REPORTING) TYPE  SONV-FLAG DEFAULT 'X'
*"     VALUE(SYNC_CHAINS) TYPE  SONV-FLAG DEFAULT 'X'
*"     VALUE(SYNC_PROG) TYPE  SONV-FLAG DEFAULT 'X'
*"     VALUE(SYNC_FUNC) TYPE  SONV-FLAG DEFAULT 'X'
*"     VALUE(SYNC_CLAS) TYPE  SONV-FLAG DEFAULT 'X'
*"     VALUE(SYNC_IOBJ) TYPE  SONV-FLAG DEFAULT 'X'
*"     VALUE(SYNC_IOBJCAT) TYPE  SONV-FLAG OPTIONAL
*"     VALUE(SYNC_AUTH) TYPE  SONV-FLAG DEFAULT 'X'
*"     VALUE(SYNC_APD) TYPE  SONV-FLAG DEFAULT 'X'
*"     VALUE(SYNC_BPC) TYPE  SONV-FLAG OPTIONAL
*"     VALUE(SYNC_WDYN) TYPE  SONV-FLAG DEFAULT 'X'
*"  TABLES
```

```
*"      T_DEVCLASS STRUCTURE  TCHLP
*"      T_LANGUAGE STRUCTURE  TCP0H
*"      T_RSDIOBJ STRUCTURE  RSDIOBJ
*"      T_RSDIOBJT STRUCTURE  RSDIOBJT
*"      T_RSDIOBCIOBJ STRUCTURE  RSDIOBCIOBJ
*"      T_RSDIOBCT STRUCTURE  RSDIOBCT
*"      T_RSBASIDOC STRUCTURE  RSBASIDOC
*"      T_RSDS STRUCTURE  RSDS
*"      T_RSDST STRUCTURE  RSDST
*"      T_RSOLTPSOURCE STRUCTURE  RSOLTPSOURCE
*"      T_RSOLTPSOURCET STRUCTURE  RSOLTPSOURCET
*"      T_RSTRANT STRUCTURE  RSTRANT
*"      T_RSIS STRUCTURE  RSIS
*"      T_RSIST STRUCTURE  RSIST
*"      T_RSKSNEW STRUCTURE  RSKSNEW
*"      T_RSKSNEWT STRUCTURE  RSKSNEWT
*"      T_RSTS STRUCTURE  RSTS
*"      T_RSDCUBET STRUCTURE  RSDCUBET
*"      T_RSDODSOT STRUCTURE  RSDODSOT
*"      T_RSQISET STRUCTURE  RSQISET
*"      T_RSQISETT STRUCTURE  RSQISETT
*"      T_RSBSPOKE STRUCTURE  RSBSPOKE
*"      T_RSBSPOKET STRUCTURE  RSBSPOKET
*"      T_RSBOHDEST STRUCTURE  RSBOHDEST
*"      T_RSBOHDESTT STRUCTURE  RSBOHDESTT
*"      T_RSPLS_ALVL STRUCTURE  RSPLS_ALVL
*"      T_RSPLS_ALVLT STRUCTURE  RSPLS_ALVLT
*"      T_RSPLF_SRV STRUCTURE  RSPLF_SRV
*"      T_RSPLF_SRVT STRUCTURE  RSPLF_SRVT
*"      T_RSPLS_SEQUENCE STRUCTURE  RSPLS_SEQUENCE
*"      T_RSPLS_SEQUENCET STRUCTURE  RSPLS_SEQUENCET
*"      T_RSPCCHAINATTR STRUCTURE  RSPCCHAINATTR
*"      T_RSPCCHAINT STRUCTURE  RSPCCHAINT
*"      T_V_REP_JOIN STRUCTURE  V_REP_JOIN
*"      T_V_REP_REUSE STRUCTURE  V_COMPDIR_COMPIC
*"      T_RSZGLOBV STRUCTURE  RSZGLOBV
*"      T_V_ELTDIR_TXT STRUCTURE  RSZELTTXT
*"      T_RSRWBINDEX STRUCTURE  RSRWBINDEX
*"      T_RSRWBINDEXT STRUCTURE  RSRWBINDEXT
*"      T_RSRWORKBOOK STRUCTURE  RSRWORKBOOK
*"      T_RSZWBTMPHEAD STRUCTURE  RSZWBTMPHEAD
*"      T_RSZWBTMPHEADTXT STRUCTURE  RSZWBTMPHEADTXT
*"      T_RSZWBTMPXREF STRUCTURE  RSZWBTMPXREF
*"      T_RSZWTEMPLATE STRUCTURE  RSZWTEMPLATE
*"      T_RSZWOBJTXT STRUCTURE  RSZWOBJTXT
*"      T_RSZWOBJXREF STRUCTURE  RSZWOBJXREF
*"      T_FUGR STRUCTURE  INFO_FUGRT
*"      T_FUNC STRUCTURE  INFO_FUNCT
*"      T_TADIR_PROG STRUCTURE  TRESN
*"      T_TRDIRT STRUCTURE  TRDIRT
*"      T_RSANT_PROCESST STRUCTURE  RSANT_PROCESST
*"      T_RSANT_PROCESS STRUCTURE  RSO_S_OBJECT_LIST
*"      T_RSDIOBC STRUCTURE  RSDIOBC
*"      T_RSISOSMAP STRUCTURE  RSISOSMAP
*"      T_RSUPDINFO STRUCTURE  RSUPDINFO
*"      T_RSDVUNI STRUCTURE  RSDVUNI
*"      T_AGR_DEFINE STRUCTURE  AGR_DEFINE
*"      T_AGR_TEXTS STRUCTURE  AGR_TEXTS
*"      T_RSDCUBEMULTI STRUCTURE  RSDCUBEMULTI
*"      T_RSQTOBJ STRUCTURE  RSQTOBJ
*"      T_RSDKYF STRUCTURE  RSDKYF
*"      T_VSEOCLASS STRUCTURE  VSEOCLASS
*"      T_WDYN STRUCTURE  TRESN
*"      T_WDY_COMPONENTT STRUCTURE  WDY_COMPONENTT
*"  EXCEPTIONS
*"      RELEASE_1_9
*"----------------------------------------------------------------------


  TYPES: BEGIN OF ty_langu,
           langu TYPE langu,
         END OF ty_langu.
  DATA: lv_laiso TYPE laiso,
        ls_langu TYPE ty_langu,
        lt_langu TYPE TABLE OF ty_langu WITH KEY langu.

  TYPES: BEGIN OF ty_clas,
            sign TYPE rssign,
```

```
            option TYPE rsoption,
            low TYPE rslow,
            high TYPE rshigh,
          END OF ty_clas.
  DATA: ls_clas TYPE ty_clas,
        lt_clas TYPE TABLE OF ty_clas.

* Structure for ABAP Reports
  TYPES: BEGIN OF ty_reports,
             progname TYPE progname,
             devclass TYPE devclass,
             udat TYPE rdir_udate,
             unam TYPE unam,
           END OF ty_reports.
  DATA: ls_tresn TYPE tresn,
        lt_reports TYPE TABLE OF ty_reports.

* Structure for Web Dynpros
  TYPES: BEGIN OF ty_wdyn,
             component TYPE wdy_component_name,
             devclass TYPE devclass,
             changedon TYPE rdir_udate,
             changedby TYPE unam,
           END OF ty_wdyn.
  DATA:   lt_wdyn TYPE TABLE OF ty_wdyn.

* Data for APDs:
  DATA: ls_rsant_process TYPE rsant_process,
        lt_rsant_process TYPE STANDARD TABLE OF rsant_process,
        ls_object_list TYPE rso_s_object_list.


  DATA: timestmp_date TYPE d, timestmp_time TYPE t, tz TYPE ttzz-tzone.

  FIELD-SYMBOLS: <fs_devclass> TYPE tchlp,
                 <fs_language> TYPE tcp0h,
                 <fs_reports> TYPE ty_reports,
                 <fs_wdyn> TYPE ty_wdyn,
                 <fs_rsant_process> TYPE rsant_process.

* Fill internal table for selected languages
LOOP AT t_language ASSIGNING <fs_language>.
      lv_laiso = <fs_language>-laiso.
      TRANSLATE lv_laiso TO UPPER CASE.
      MODIFY t_language FROM lv_laiso.
ENDLOOP.

SELECT spras FROM t002 INTO TABLE lt_langu
  FOR ALL ENTRIES IN t_language
  WHERE laiso = t_language-laiso.


* Fill internal table for selected packages
  LOOP AT t_devclass ASSIGNING <fs_devclass>.
    ls_clas-sign = 'I'.
    ls_clas-option = 'CP'.
    ls_clas-low = <fs_devclass>-devclass.
    APPEND ls_clas TO lt_clas.
  ENDLOOP.


  IF timestmp IS INITIAL.
    timestmp = 0.
  ENDIF.

  tz = 'UTC'. "sy-zonlo.
  CONVERT TIME STAMP timestmp TIME ZONE tz
        INTO DATE timestmp_date TIME timestmp_time.


  IF sync_workbench = 'X' OR sync_reporting = 'X'.
* CUBE, only texts (RSDCUBE is synchr. from application)
    SELECT rsdcube~infocube t~langu t~txtsh t~txtlg FROM rsdcube
    INNER JOIN rsdcubet AS t
    ON rsdcube~infocube = t~infocube
    AND rsdcube~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsdcubet
      FOR ALL ENTRIES IN lt_langu
```

```
      WHERE
        rsdcube~objvers = 'A' AND
        rsdcube~cubetype <> 'A' AND
        rsdcube~infocube NOT LIKE '/CPMB/%' AND
        rsdcube~timestmp >= timestmp AND
        t~langu = lt_langu-langu.

* ODSO, only texts (RSDODSO is synchr. from application)
    SELECT rsdodso~odsobject t~langu t~txtsh t~txtlg FROM rsdodso
      INNER JOIN rsdodsot AS t
     ON rsdodso~odsobject = t~odsobject
     AND rsdodso~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsdodsot
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rsdodso~objvers = 'A' AND
        rsdodso~timestmp >= timestmp AND
        t~langu = lt_langu-langu.

* ISET
    SELECT infoset timestmp tstpnm infoarea FROM rsqiset
      INTO CORRESPONDING FIELDS OF TABLE t_rsqiset WHERE
        objvers = 'A' AND
        timestmp >= timestmp.

    SELECT rsqiset~infoset t~langu t~txtsh t~txtlg FROM rsqiset
      INNER JOIN rsqisett AS t
     ON rsqiset~infoset = t~infoset
     AND rsqiset~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsqisett
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rsqiset~objvers = 'A' AND
        rsqiset~timestmp >= timestmp AND
        t~langu = lt_langu-langu.

* ALVL
    SELECT aggrlevel infoprov timestmp tstpnm infoarea FROM rspls_alvl
      INTO CORRESPONDING FIELDS OF TABLE t_rspls_alvl WHERE
        objvers = 'A' AND
        timestmp >= timestmp.

    SELECT rspls_alvl~aggrlevel t~langu t~txtsh t~txtlg FROM rspls_alvl
      INNER JOIN rspls_alvlt AS t
     ON rspls_alvl~aggrlevel = t~aggrlevel
     AND rspls_alvl~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rspls_alvlt
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rspls_alvl~objvers = 'A' AND
        rspls_alvl~timestmp >= timestmp AND
        t~langu = lt_langu-langu.

* PLSE
    SELECT srvnm infoprov timestmp tstpnm FROM rsplf_srv
        INTO CORRESPONDING FIELDS OF TABLE t_rsplf_srv WHERE
          objvers = 'A' AND
          timestmp >= timestmp.

    SELECT rsplf_srv~srvnm t~langu t~txtlg FROM rsplf_srv
      INNER JOIN rsplf_srvt AS t
     ON rsplf_srv~srvnm = t~srvnm
     AND rsplf_srv~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsplf_srvt
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rsplf_srv~objvers = 'A' AND
        rsplf_srv~timestmp >= timestmp AND
        t~langu = lt_langu-langu.

* PLSQ
    SELECT seqnm timestmp tstpnm FROM rspls_sequence
        INTO CORRESPONDING FIELDS OF TABLE t_rspls_sequence WHERE
          objvers = 'A' AND
          timestmp >= timestmp.

    SELECT rspls_sequence~seqnm t~langu t~txtlg FROM rspls_sequence
      INNER JOIN rspls_sequencet AS t
```

```
      ON rspls_sequence~seqnm = t~seqnm
      AND rspls_sequence~objvers = t~objvers
       INTO CORRESPONDING FIELDS OF TABLE t_rspls_sequencet
       FOR ALL ENTRIES IN lt_langu
       WHERE
         rspls_sequence~objvers = 'A' AND
         rspls_sequence~timestmp >= timestmp AND
         t~langu = lt_langu-langu.

* SPOK (InfoSpokes)
    SELECT infospoke timestmp tstpnm ohsource FROM rsbspoke
      INTO CORRESPONDING FIELDS OF TABLE t_rsbspoke WHERE
        objvers = 'A' AND
        timestmp >= timestmp.

    SELECT rsbspoke~infospoke t~langu t~txtsh t~txtlg FROM rsbspoke
      INNER JOIN rsbspoket AS t
     ON rsbspoke~infospoke = t~infospoke
     AND rsbspoke~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsbspoket
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rsbspoke~objvers = 'A' AND
        rsbspoke~timestmp >= timestmp AND
        t~langu = lt_langu-langu.

* DEST (Destinations)
    SELECT ohdest desttype logsys timestmp tstpnm new_ohd infoarea
       FROM rsbohdest
      INTO CORRESPONDING FIELDS OF TABLE t_rsbohdest WHERE
        objvers = 'A' AND
        new_ohd = 'X' AND
        timestmp >= timestmp.

    SELECT rsbohdest~ohdest t~langu t~txtsh t~txtlg FROM rsbohdest
      INNER JOIN rsbohdestt AS t
     ON rsbohdest~ohdest = t~ohdest
     AND rsbohdest~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsbohdestt
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rsbohdest~objvers = 'A' AND
        new_ohd = 'X' AND
        rsbohdest~timestmp >= timestmp AND
        t~langu = lt_langu-langu.

  ENDIF.


  IF sync_apd = 'X'.
* ANPR (Analysis Process)
    SELECT process tstpnm timestmp appl FROM rsant_process
         INTO CORRESPONDING FIELDS OF TABLE lt_rsant_process WHERE
           objvers = 'A' AND
           timestmp >= timestmp.
    LOOP AT lt_rsant_process ASSIGNING <fs_rsant_process>.
      ls_object_list-tlogo = 'ANPR'.
      ls_object_list-objnm = <fs_rsant_process>-process.
      ls_object_list-tstpnm = <fs_rsant_process>-tstpnm.
      ls_object_list-timestmp = <fs_rsant_process>-timestmp.
      ls_object_list-txtlg = <fs_rsant_process>-appl.
      APPEND ls_object_list TO t_rsant_process.
    ENDLOOP.

    SELECT rsant_process~process t~spras t~txtsh t~txtlg
        FROM rsant_process
        INNER JOIN rsant_processt AS t
       ON rsant_process~process = t~process
       AND rsant_process~objvers = t~objvers
        INTO CORRESPONDING FIELDS OF TABLE t_rsant_processt
        FOR ALL ENTRIES IN lt_langu
        WHERE
          rsant_process~objvers = 'A' AND
          rsant_process~timestmp >= timestmp AND
          t~spras = lt_langu-langu.
  ENDIF.

*Synch InfoObject-Catalogs
```

```
  IF sync_iobjcat = 'X'.
* Catalogs
    SELECT * FROM rsdiobc
      INTO TABLE t_rsdiobc WHERE
      timestmp >= timestmp AND
        objvers = 'A'.


* Texts for Catalogs
    SELECT rsdiobc~infoobjcat t~langu t~txtsh t~txtlg FROM rsdiobc
        INNER JOIN rsdiobct AS t
       ON rsdiobc~infoobjcat = t~infoobjcat
       AND rsdiobc~objvers = t~objvers
        INTO CORRESPONDING FIELDS OF TABLE t_rsdiobct
        FOR ALL ENTRIES IN lt_langu
        WHERE
          rsdiobc~objvers = 'A' AND
          rsdiobc~timestmp >= timestmp AND
          t~langu = lt_langu-langu.
  ENDIF.


  IF sync_iobj = 'X' OR sync_workbench = 'X'.
    "Necessary when Workbench is synched (for Char-InfoProvider)!!
* IOBJ
    SELECT iobjnm iobjtp timestmp tstpnm FROM rsdiobj
    INTO CORRESPONDING FIELDS OF TABLE t_rsdiobj WHERE
      objvers = 'A' AND
      timestmp >= timestmp.

    SELECT iobjnm langu txtsh txtlg FROM rsdiobjv
         INTO CORRESPONDING FIELDS OF TABLE t_rsdiobjt
         FOR ALL ENTRIES IN lt_langu
         WHERE
           objvers = 'A' AND
           langu = lt_langu-langu AND
           timestmp >= timestmp.

*Characteristics are not synchr. here because in BW 7.4 the View
*RSDVCHA can't be added as table parameter (contains SSTRING)

  ENDIF.

  IF sync_iobj = 'X'.
* Details for KeyFigures
    SELECT * FROM rsdkyf AS k
              INNER JOIN rsdiobj
              ON k~kyfnm = rsdiobj~iobjnm AND
              rsdiobj~objvers = k~objvers
              INTO CORRESPONDING FIELDS OF TABLE t_rsdkyf
              WHERE
                rsdiobj~timestmp >= timestmp AND
                rsdiobj~objvers = 'A'.

*Time Characteristics are not synchr. here because in BW 7.4 the
*View RSDVTIM can't be added as table parameter (contains SSTRING)


* Details for Units
    SELECT * FROM rsdvuni AS u
              INNER JOIN rsdiobj
              ON u~uninm = rsdiobj~iobjnm AND
              rsdiobj~objvers = u~objvers
              INTO CORRESPONDING FIELDS OF TABLE t_rsdvuni
              WHERE
                rsdiobj~timestmp >= timestmp AND
                rsdiobj~objvers = 'A'.

  ENDIF.



  IF sync_reporting = 'X' OR sync_workbench = 'X'.
    "in case of workbench sync we also need to sync Queries because
    "Queries can be used in new BW releases also in Transformations!
* Queries must be read from v_rep_join because GENUNIID is needed!
    SELECT * FROM v_rep_join
       INTO TABLE t_v_rep_join WHERE
```

```
          objvers = 'A' AND
          deftp = 'REP' AND
          ( timestmp >= timestmp OR lastused >= timestmp ).

* Get all texts to reusable reporting components
    SELECT t~eltuid t~langu t~txtsh t~txtlg FROM rszcompdir
         INNER JOIN rszelttxt AS t
         ON rszcompdir~compuid = t~eltuid
         AND rszcompdir~objvers = t~objvers
         INTO CORRESPONDING FIELDS OF TABLE t_v_eltdir_txt
         FOR ALL ENTRIES IN lt_langu
      WHERE
          rszcompdir~objvers = 'A' AND
          rszcompdir~timestmp >= timestmp AND
          t~langu = lt_langu-langu.
  ENDIF.

  IF sync_reporting = 'X'.
* Reusable KeyFigures, Structures and Filters
* V_COMPDIR_COMPIC delivers duplicates in case of diff. languages!
* better use V_CMP_JOIN
    SELECT DISTINCT compuid infocube compid
       version deftp timestmp tstpnm
    FROM v_compdir_compic
    INTO CORRESPONDING FIELDS OF TABLE t_v_rep_reuse
      WHERE
      objvers = 'A' AND
      timestmp >= timestmp AND (
      deftp = 'SOB' OR
      deftp = 'STR' OR
      deftp = 'SEL' OR
      deftp = 'CKF' ).

* Variables
    SELECT * FROM rszglobv INTO TABLE t_rszglobv
    WHERE
           objvers = 'A' AND
           timestmp >= timestmp.

* XLWB (Workbooks)
    SELECT * FROM rsrwbindex INTO TABLE t_rsrwbindex
    WHERE objvers = 'A' AND
          timestmp >= timestmp.

    SELECT rsrwbindex~workbookid t~langu t~title FROM rsrwbindex
          INNER JOIN rsrwbindext AS t
         ON rsrwbindex~workbookid = t~workbookid
         AND rsrwbindex~objvers = t~objvers
          INTO CORRESPONDING FIELDS OF TABLE t_rsrwbindext
          FOR ALL ENTRIES IN lt_langu
          WHERE
            rsrwbindex~objvers = 'A' AND
            rsrwbindex~timestmp >= timestmp AND
            t~langu = lt_langu-langu.


* BTMP (7.x WebTemplates)
    SELECT * FROM rszwbtmphead INTO TABLE t_rszwbtmphead
    WHERE objvers = 'A' AND
         timestmp >= timestmp.

    SELECT t~objid t~langu t~txtlg FROM rszwbtmphead
           INNER JOIN rszwbtmpheadtxt AS t
          ON rszwbtmphead~objid = t~objid
          AND rszwbtmphead~objvers = t~objvers
           INTO CORRESPONDING FIELDS OF TABLE t_rszwbtmpheadtxt
           FOR ALL ENTRIES IN lt_langu
           WHERE
             rszwbtmphead~objvers = 'A' AND
             rszwbtmphead~timestmp >= timestmp AND
             t~langu = lt_langu-langu.

* TMPL (3.x WebTemplates)
    SELECT * FROM rszwtemplate INTO TABLE t_rszwtemplate
     WHERE objvers = 'A' AND
          timestmp >= timestmp.

    SELECT t~objid t~langu t~txtlg FROM rszwtemplate
```

```
           INNER JOIN rszwobjtxt AS t
          ON rszwtemplate~tmplid = t~objid
          AND rszwtemplate~objvers = t~objvers
           INTO CORRESPONDING FIELDS OF TABLE t_rszwobjtxt
           FOR ALL ENTRIES IN lt_langu
           WHERE
             rszwtemplate~objvers = 'A' AND
             rszwtemplate~timestmp >= timestmp AND
             t~langu = lt_langu-langu.
  ENDIF.


  IF sync_workbench = 'X'.
* Source Systems
    SELECT * FROM rsbasidoc
     INTO TABLE t_rsbasidoc
    WHERE timestmp >= timestmp.


* ISFS (3.x DataSources)
*    SELECT * FROM rsoltpsource
*      INTO CORRESPONDING FIELDS OF TABLE rsoltpsource WHERE
*        objvers = 'A' AND
*        timestmp >= timestmp.

    SELECT s~oltpsource s~logsys s~type s~tstpnm s~timestmp
      FROM rsoltpsource AS s
      INNER JOIN rsisosmap AS m
      ON s~oltpsource = m~oltpsource
        AND s~logsys = m~logsys
        AND s~objvers = m~objvers
    INTO CORRESPONDING FIELDS OF TABLE t_rsoltpsource
       WHERE
      s~objvers = 'A' AND
      s~timestmp >= timestmp.


    SELECT rsoltpsource~oltpsource rsoltpsource~logsys t~langu
           t~txtsh t~txtlg
       FROM rsoltpsource
       INNER JOIN rsoltpsourcet AS t
      ON rsoltpsource~oltpsource = t~oltpsource
      AND rsoltpsource~logsys = t~logsys
      AND rsoltpsource~objvers = t~objvers
      INNER JOIN rsisosmap AS m "to get only DS used in 3.x Data Flows
      ON rsoltpsource~oltpsource = m~oltpsource
        AND rsoltpsource~logsys = m~logsys
        AND rsoltpsource~objvers = m~objvers
         INTO CORRESPONDING FIELDS OF TABLE t_rsoltpsourcet
         FOR ALL ENTRIES IN lt_langu
         WHERE
           rsoltpsource~objvers = 'A' AND
           rsoltpsource~timestmp >= timestmp AND
           t~langu = lt_langu-langu.


* RSDS (7.x DS); Always full sync because this table is only relevant
* for getting version and date of change, the DataSources
* will be received  from table RSTRAN
    SELECT datasource logsys type applnm exstructure delta
           timestmp tstpnm
      FROM rsds
      INTO CORRESPONDING FIELDS OF TABLE t_rsds WHERE
        objvers = 'A'." AND timestmp >= timestmp.

    SELECT rsds~datasource rsds~logsys t~langu t~txtsh t~txtlg
      FROM rsds
      INNER JOIN rsdst AS t
     ON rsds~datasource = t~datasource
     AND rsds~logsys = t~logsys
     AND rsds~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsdst
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rsds~objvers = 'A' AND
        rsds~timestmp >= timestmp AND
        t~langu = lt_langu-langu.
```

```
* RSIS (3.x InfoSources)
    SELECT isource comstru timestmp tstpnm FROM rsis
      INTO CORRESPONDING FIELDS OF TABLE t_rsis WHERE
        objvers = 'A' AND
        timestmp >= timestmp.

    SELECT rsis~isource t~langu t~txtsh t~txtlg FROM rsis
      INNER JOIN rsist AS t
     ON rsis~isource = t~isource
     AND rsis~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsist
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rsis~objvers = 'A' AND
        rsis~timestmp >= timestmp AND
        t~langu = lt_langu-langu.


* TRCS (7.x InfoSources)
    SELECT isource timestmp tstpnm FROM rsksnew
        INTO CORRESPONDING FIELDS OF TABLE t_rsksnew WHERE
          objvers = 'A' AND
          timestmp >= timestmp.

    SELECT rsksnew~isource t~langu t~txtlg FROM rsksnew
      INNER JOIN rsksnewt AS t
     ON rsksnew~isource = t~isource
     AND rsksnew~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsksnewt
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rsksnew~objvers = 'A' AND
        rsksnew~timestmp >= timestmp AND
        t~langu = lt_langu-langu.

* 3.x Transfer Structures
    SELECT * FROM rsts
        INTO TABLE t_rsts WHERE
          objvers = 'A' AND
          timestmp >= timestmp.



* TRFN (7.x Transformations), only texts
    SELECT rstran~tranid t~langu t~txtsh t~txtlg FROM rstran
      INNER JOIN rstrant AS t
     ON rstran~tranid = t~tranid
     AND rstran~objvers = t~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rstrant
      FOR ALL ENTRIES IN lt_langu
      WHERE
        rstran~objvers = 'A' AND
        rstran~timestmp >= timestmp AND
        t~langu = lt_langu-langu.

* RSISOSMAP (Mapping between InfoSources and OLTP Sources, 3.x)
    "the timestamp in RSISOSMAP is not updated in SAP, so join on RSTS
    SELECT * FROM rsisosmap
      INNER JOIN rsts
      ON rsisosmap~transtru = rsts~transtru
      AND rsisosmap~objvers = rsts~objvers
      INTO CORRESPONDING FIELDS OF TABLE t_rsisosmap
      WHERE
        rsisosmap~objvers = 'A' AND
        rsts~timestmp >= timestmp.


* RSUPDINFO (Mapping Infos for Update Rules, 3.x)
    SELECT * FROM rsupdinfo
            INTO TABLE t_rsupdinfo WHERE
              objvers = 'A' AND
              timestmp >= timestmp.

* RSDCUBEMULTI
    SELECT m~infocube m~posit m~partcube FROM rsdcube AS c
    INNER JOIN rsdcubemulti AS m
```

```
      ON m~infocube = c~infocube
      AND m~objvers = c~objvers
       INTO CORRESPONDING FIELDS OF TABLE t_rsdcubemulti WHERE
         c~objvers = 'A' AND
         m~infocube NOT LIKE '/CPMB/%' AND
         c~timestmp >= timestmp.

* RSQTOBJ (Table Objects in the InfoSet)
    SELECT t~infoset t~talias t~tname t~ttype FROM rsqiset AS i
    INNER JOIN rsqtobj AS t
     ON i~infoset = t~infoset
     AND i~objvers = t~objvers
       INTO CORRESPONDING FIELDS OF TABLE t_rsqtobj WHERE
        i~objvers = 'A' AND
        i~timestmp >= timestmp AND
        t~ttype <> ''.


  ENDIF.


  IF sync_chains = 'X'.
* RSPC (Process Chains) -> no real date of change available,
* because running a chain activates it automatically!
    IF sync_bpc <> 'X'."BPC Module is not lic., exclude BPC chains
      SELECT chain_id timestmp tstpnm applnm FROM rspcchainattr
          INTO CORRESPONDING FIELDS OF TABLE t_rspcchainattr WHERE
            objvers = 'A' AND
            chain_id NOT LIKE '/CPMB/%' AND
            timestmp >= timestmp.

      SELECT rspcchainattr~chain_id t~langu t~txtlg FROM rspcchainattr
        INNER JOIN rspcchaint AS t
       ON rspcchainattr~chain_id = t~chain_id
       AND rspcchainattr~objvers = t~objvers
        INTO CORRESPONDING FIELDS OF TABLE t_rspcchaint
        FOR ALL ENTRIES IN lt_langu
        WHERE
          rspcchainattr~objvers = 'A' AND
          rspcchainattr~chain_id NOT LIKE '/CPMB/%' AND
          rspcchainattr~timestmp >= timestmp AND
          t~langu = lt_langu-langu.
    ELSE.
      SELECT chain_id timestmp tstpnm applnm FROM rspcchainattr
                INTO CORRESPONDING FIELDS OF TABLE t_rspcchainattr
                WHERE objvers = 'A' AND timestmp >= timestmp.

      SELECT rspcchainattr~chain_id t~langu t~txtlg FROM rspcchainattr
        INNER JOIN rspcchaint AS t
       ON rspcchainattr~chain_id = t~chain_id
       AND rspcchainattr~objvers = t~objvers
        INTO CORRESPONDING FIELDS OF TABLE t_rspcchaint
        FOR ALL ENTRIES IN lt_langu
        WHERE
          rspcchainattr~objvers = 'A' AND
          rspcchainattr~timestmp >= timestmp AND
          t~langu = lt_langu-langu.
    ENDIF.

  ENDIF.


  IF sync_func = 'X'.
* FUNC (Function Modules by dev-classes) View: INFO_FUNCT
* No language selection because elements might be missing if no
* description maintained in the chosen language
    SELECT funcname spras area pname include fmode devclass stext
          FROM info_funct
           INTO CORRESPONDING FIELDS OF TABLE t_func
            WHERE
               info_funct~devclass IN lt_clas.

*  select name sdate stime from trdir
*   into corresponding fields of table func where
*   SUBC = 'I' and
*  SDATE = timestmp_date and
*  stime >= timestmp_time.
```

```
* FUGR (Function Groups + texts by dev-classes)
* No language selection because elements might be missing if no
* description maintained in the chosen language
    SELECT * FROM info_fugrt
       INTO TABLE t_fugr WHERE
         devclass IN lt_clas.

  ENDIF.


  IF sync_prog = 'X'.
* PROG (ABAP Reports)
    SELECT tadir~obj_name tadir~devclass
           trdir~udat trdir~unam
     FROM tadir
      INNER JOIN trdir ON
       tadir~obj_name = trdir~name
        INTO TABLE lt_reports
          WHERE
            tadir~pgmid = 'R3TR' AND
            tadir~object = 'PROG' AND
            tadir~devclass IN lt_clas AND
            trdir~udat >= timestmp_date.
    LOOP AT lt_reports ASSIGNING <fs_reports>.
      ls_tresn-obj_namelo = <fs_reports>-progname.
      ls_tresn-devclass = <fs_reports>-devclass.
      ls_tresn-moddate = <fs_reports>-udat.
      ls_tresn-author = <fs_reports>-unam.
      APPEND ls_tresn TO t_tadir_prog.
    ENDLOOP.

* PROG (Texts for ABAP Reports)
    SELECT trdirt~name trdirt~sprsl trdirt~text FROM trdirt
    INNER JOIN tadir ON
      trdirt~name = tadir~obj_name
      INNER JOIN trdir ON
        trdirt~name = trdir~name
         INTO TABLE t_trdirt
          FOR ALL ENTRIES IN lt_langu
           WHERE
            sprsl = lt_langu-langu AND
            tadir~pgmid = 'R3TR' AND
            tadir~object = 'PROG' AND
            tadir~devclass IN lt_clas AND
            trdir~udat >= timestmp_date.
  ENDIF.

  IF sync_wdyn = 'X'.
* WDYN (Web Dynpros)
    SELECT tadir~obj_name tadir~devclass
           wdy_component~changedon wdy_component~changedby
     FROM tadir
      INNER JOIN wdy_component ON
       tadir~obj_name = wdy_component~component_name
        INTO TABLE lt_wdyn
          WHERE
            tadir~pgmid = 'R3TR' AND
            tadir~object = 'WDYN' AND
            tadir~delflag <> 'X' AND
            tadir~devclass IN lt_clas AND
            wdy_component~version = 'A' AND
            wdy_component~type = '0' AND
            wdy_component~changedon >= timestmp_date.
    LOOP AT lt_wdyn ASSIGNING <fs_wdyn>.
      ls_tresn-obj_namelo = <fs_wdyn>-component.
      ls_tresn-devclass = <fs_wdyn>-devclass.
      ls_tresn-moddate = <fs_wdyn>-changedon.
      ls_tresn-author = <fs_wdyn>-changedby.
      APPEND ls_tresn TO t_wdyn.
    ENDLOOP.

* WDYN (Texts for Web Dynpros)
    SELECT  wdy_componentt~component_name wdy_componentt~langu
            wdy_componentt~description
      FROM wdy_componentt
    INNER JOIN tadir ON
      wdy_componentt~component_name = tadir~obj_name
      INNER JOIN wdy_component ON
```

```
        wdy_componentt~component_name = wdy_component~component_name
         INTO TABLE t_wdy_componentt
          FOR ALL ENTRIES IN lt_langu
           WHERE
            langu = lt_langu-langu AND
            wdy_component~version = 'A' AND
            wdy_component~type = '0' AND
            tadir~pgmid = 'R3TR' AND
            tadir~object = 'WDYN' AND
            tadir~delflag <> 'X' AND
            tadir~devclass IN lt_clas AND
            wdy_component~changedon >= timestmp_date.
  ENDIF.

  IF sync_clas = 'X'.
* CLAS (ABAP Classes), sync texts and tech. info together
    SELECT vseoclass~clsname vseoclass~langu vseoclass~descript
           vseoclass~changedby vseoclass~changedon tadir~devclass
       FROM vseoclass INNER JOIN tadir
        ON vseoclass~clsname = tadir~obj_name
         INTO CORRESPONDING FIELDS OF TABLE t_vseoclass
           WHERE
            tadir~pgmid = 'R3TR' AND
            tadir~object = 'CLAS' AND
            tadir~devclass IN lt_clas AND
            vseoclass~version = '1' AND
            vseoclass~changedon >= timestmp_date.
  ENDIF.


  IF sync_auth = 'X'.
    SELECT mandt agr_name parent_agr change_usr change_tim change_dat
     FROM agr_define
      INTO CORRESPONDING FIELDS OF TABLE t_agr_define
      WHERE agr_name NOT LIKE 'SAP%' AND
            change_dat >= timestmp_date.

    SELECT agr_texts~mandt agr_texts~agr_name agr_texts~spras
           agr_texts~line agr_texts~text
      FROM agr_texts
      INNER JOIN agr_define ON
       agr_define~mandt = agr_texts~mandt AND
       agr_define~agr_name = agr_texts~agr_name
      INTO TABLE t_agr_texts
       FOR ALL ENTRIES IN lt_langu
         WHERE
          spras = lt_langu-langu AND
          agr_texts~agr_name NOT LIKE 'SAP%' AND
          line = '00000' AND
          agr_define~change_dat >= timestmp_date.
  ENDIF.


ENDFUNCTION.
```

8 Function Module Z_RFC_FUNCTION_DELETE - Delete Function Module (for Updating FMs)

8.1 General Information

System	BI2
Function Module	Z_RFC_FUNCTION_DELETE, Delete Function Module (for Updating FMs)
Function Pool	Z_DP
Remote	Yes
Last changed by	ADUERRSTEIN
Last change (timestamp)	09/03/2018
Timestamp of documentation	11/08/2020 16:29:51

8.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text

FUNCNAME	TYPE	FUNCNAME			X	Funktionsname

8.3 Exceptions

Exception	Short text
ERROR_MESSAGE	
RELEASE_1_0	

8.4 Source code Function module

```
FUNCTION z_rfc_function_delete.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(FUNCNAME) TYPE  FUNCNAME
*"  EXCEPTIONS
*"      ERROR_MESSAGE
*"      RELEASE_1_0
*"----------------------------------------------------------------------
  CALL FUNCTION 'FUNCTION_DELETE'
    EXPORTING
      funcname      = funcname
    EXCEPTIONS
      error_message = 1
      OTHERS        = 2.
  IF sy-subrc = 1.
    RAISE ERROR_MESSAGE.
  ENDIF.
ENDFUNCTION.
```

9 Function Module Z_RFC_GET_DTP_DETAILS - Read DTP filter via RFC

9.1 General Information

System	BI2
Function Module	Z_RFC_GET_DTP_DETAILS, Read DTP filter via RFC
Function Pool	Z_DP
Remote	Yes
Last changed by	TSCHMIDT
Last change (timestamp)	08/12/2015
Timestamp of documentation	11/08/2020 16:29:54

9.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
I_DTP	TYPE	RSBKDTPNM			X	DTP ID

9.3 Export Parameter

Parameter		Associated Type	Pass	Short text
E_MAXSIZE	TYPE	RSBKMAXSIZE	X	DTP-Extraktion: Paketgröße

9.4 Tables

Parameter		Associated Type	Opt.	Short text
DATA_SEL	LIKE	RSSELECT		Selections in DTP
DATA_RULE	LIKE	MCH_S_SOURCECODE		SourceCode of DTP-Routines (RSLDPRULE)
DATA_VAR	LIKE	MCH_VAR_SELECT		Variables in DTP-Selections
DATA_GROUP	LIKE	RSBK_S_FIELDS_KEYFL		Semantic Groupfields

9.5 Exceptions

Exception	Short text
RELEASE_1_2	

9.6 Source code Function module

```
FUNCTION z_rfc_get_dtp_details.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_DTP) TYPE  RSBKDTPNM
*"  EXPORTING
*"     VALUE(E_MAXSIZE) TYPE  RSBKMAXSIZE
*"  TABLES
*"      DATA_SEL STRUCTURE  RSSELECT
*"      DATA_RULE STRUCTURE  MCH_S_SOURCECODE
*"      DATA_VAR STRUCTURE  MCH_VAR_SELECT
*"      DATA_GROUP STRUCTURE  RSBK_S_FIELDS_KEYFL
*"  EXCEPTIONS
*"      RELEASE_1_2
*"----------------------------------------------------------------------


  DATA l_r_rsbk_dtp TYPE REF TO cl_rsbk_dtp.
  DATA l_r_rsbc_filter TYPE REF TO cl_rsbc_filter.
  DATA l_r_rsbc_error_handler_tpl
         TYPE REF TO cl_rsbc_error_handler_tpl.

  DATA: l_t_groupfields  TYPE rsbk_tx_fields_keyfl,
        l_th_groupfields TYPE HASHED TABLE OF rsbk_sx_fields_keyfl
                          WITH UNIQUE KEY segid.
  DATA ls_group       TYPE rsbk_s_fields_keyfl.
  DATA ls_seltab      TYPE rsbk_s_select.
  DATA ls_ruletab     TYPE mch_s_sourcecode.
  DATA ls_vartab      TYPE mch_var_select.
  DATA ls_groupfields TYPE rsbk_sx_fields_keyfl.
  DATA lv_sel         TYPE rsselect.
  DATA lv_rule        TYPE mch_s_sourcecode.
  DATA lv_var         TYPE mch_var_select.
  DATA lv_group       TYPE rsbk_s_fields_keyfl.
  DATA lv_maxsize        TYPE rsbkmaxsize.

* ==== Get Filter details ====
  TRY.
      l_r_rsbk_dtp = cl_rsbk_dtp=>factory( i_dtp ).
      l_r_rsbc_filter =
       l_r_rsbk_dtp->if_rsbk_dtp_display~get_obj_ref_filter( ).
    CATCH cx_rs_access_error.
  ENDTRY.

* ==== Get Semantic Groups details ====
  CALL METHOD l_r_rsbk_dtp->if_rsbk_dtp_display~get_groupfields
    IMPORTING
      e_t_groupfields = l_t_groupfields.
  INSERT LINES OF l_t_groupfields INTO TABLE l_th_groupfields.


* Fill result table for selections:
  LOOP AT l_r_rsbc_filter->n_t_seltab INTO ls_seltab.

    MOVE ls_seltab-field TO lv_sel-fieldnm.
    MOVE ls_seltab-sign TO lv_sel-sign.
    MOVE ls_seltab-option TO lv_sel-option.
    MOVE ls_seltab-low TO lv_sel-low.
    MOVE ls_seltab-high TO lv_sel-high.
    APPEND lv_sel TO data_sel.

  ENDLOOP.

* Fill result table for ABAP-Routines in selections
  LOOP AT l_r_rsbc_filter->n_t_dtprule INTO ls_ruletab.

    MOVE-CORRESPONDING ls_ruletab TO lv_rule.
    APPEND lv_rule TO data_rule.

  ENDLOOP.
```

```
* Fill result table for Variables in selections
  LOOP AT l_r_rsbc_filter->n_t_varseltab INTO ls_vartab.

    MOVE-CORRESPONDING ls_vartab TO lv_var.
    APPEND lv_var TO data_var.

  ENDLOOP.

* Fill result table for Semantic Groups
  LOOP AT l_th_groupfields INTO ls_groupfields.
    LOOP AT ls_groupfields-t_fields INTO ls_group.

      MOVE ls_group-fieldname TO lv_group-fieldname.
      MOVE ls_group-txtlg TO lv_group-txtlg.
      APPEND lv_group TO data_group.

    ENDLOOP.
  ENDLOOP.

* Get Max. Package Size
  CALL METHOD l_r_rsbk_dtp->get_maxsize
    RECEIVING
      r_maxsize = lv_maxsize.

e_maxsize = lv_maxsize.

ENDFUNCTION.
```

10 Function Module Z_RFC_GET_STRING - Read STRING fields

10.1 General Information

System	BI2
Function Module	Z_RFC_GET_STRING, Read STRING fields
Function Pool	Z_DP
Remote	Yes
Last changed by	ADUERRSTEIN
Last change (timestamp)	15/03/2018
Timestamp of documentation	11/08/2020 16:29:57

10.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
I_TABLE	TYPE	DD02L-TABNAME			X	
I_FIELD	TYPE	FELD_NAME		X	X	
I_RAWSTRING	TYPE	SONV-FLAG		X	X	
I_SCAN_STRING	TYPE	CHAR80		X	X	
I_SCAN_FIELD	TYPE	FELD_NAME		X	X	
I_ROWCOUNT	TYPE	SOID-ACCNT	0	X	X	
I_DECRYPTION	TYPE	SONV-FLAG	'X'	X	X	

10.3 Export Parameter

Parameter		Associated Type	Pass	Short text
E_STRING	TYPE	STRING	X	
E_STRINGTAB	TYPE	STRINGTAB	X	
E_RC	TYPE	INT4	X	
E_MSG	TYPE	STRING	X	

10.4 Tables

Parameter		Associated Type	Opt.	Short text
T_FIELDS	LIKE	RFC_DB_FLD		

T_OPTIONS	LIKE	RTXTLDAT		

10.5 Exceptions

Exception	Short text
RELEASE_1_7	
NOT_AUTHORIZED	
TABLE_NOT_AVAILABLE	

10.6 Source code Function module

```
FUNCTION Z_RFC_GET_STRING.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(I_TABLE) TYPE  DD02L-TABNAME
*"     VALUE(I_FIELD) TYPE  FELD_NAME OPTIONAL
*"     VALUE(I_RAWSTRING) TYPE  SONV-FLAG OPTIONAL
*"     VALUE(I_SCAN_STRING) TYPE  CHAR80 OPTIONAL
*"     VALUE(I_SCAN_FIELD) TYPE  FELD_NAME OPTIONAL
*"     VALUE(I_ROWCOUNT) TYPE  SOID-ACCNT DEFAULT 0
*"     VALUE(I_DECRYPTION) TYPE  SONV-FLAG DEFAULT 'X'
*"  EXPORTING
*"     VALUE(E_STRING) TYPE  STRING
*"     VALUE(E_STRINGTAB) TYPE  STRINGTAB
*"     VALUE(E_RC) TYPE  INT4
*"     VALUE(E_MSG) TYPE  STRING
*"  TABLES
*"      T_FIELDS STRUCTURE  RFC_DB_FLD
*"      T_OPTIONS STRUCTURE  RTXTLDAT
*"  EXCEPTIONS
*"      RELEASE_1_7
*"      NOT_AUTHORIZED
*"      TABLE_NOT_AVAILABLE
*"----------------------------------------------------------------------
  "Check Auth.
  CALL FUNCTION 'VIEW_AUTHORITY_CHECK'
    EXPORTING
      view_action                    = 'S'
      view_name                      = i_table
    EXCEPTIONS
      no_authority                   = 2
      no_clientindependent_authority = 2
      no_linedependent_authority     = 2
      OTHERS                         = 1.

  IF sy-subrc = 2.
    RAISE not_authorized.
  ELSEIF sy-subrc = 1.
    RAISE table_not_available.
  ENDIF.

  DATA:
    lr_error     TYPE REF TO cx_root,
    lr_dataref   TYPE REF TO data,
    lv_rawstring TYPE rswr_data_xstring,
    lv_zip       TYPE c LENGTH 1.

* Variables
  DATA:
    lv_string     TYPE string,
    lv_decrypt    TYPE string,
    lv_temp       TYPE string,
    ls_dfies      TYPE dfies,
    lt_dfields    TYPE ddfields,
    lt_alldfields TYPE ddfields,
    lt_stringtab  TYPE stringtab, "temp. result table
    lv_skip       TYPE c LENGTH 1. "Flag for skipping dataset insert
  FIELD-SYMBOLS:
    <lt_tab>  TYPE ANY TABLE,
    <ls_line> TYPE any,
    <lv_any>  TYPE any.


  IF i_field IS INITIAL.
*catch any exception and pass message along to caller
```

```
    TRY.

*Get meta data for table
        CALL FUNCTION 'DDIF_NAMETAB_GET'
          EXPORTING
            tabname   = i_table
          TABLES
            dfies_tab = lt_alldfields
          EXCEPTIONS
            OTHERS    = 2.

        IF sy-subrc <> 0.
          e_rc = 4.
          e_msg = 'TABLE NOT FOUND, PLEASE TRY AGAIN.'.
          RETURN.
        ENDIF.

        IF t_fields[] IS INITIAL.
*No restrictions provided; use all fields
          lt_dfields = lt_alldfields.
        ELSE.
*Verify that all fieldnames specified exist in table
          LOOP AT t_fields ASSIGNING <lv_any>.
            READ TABLE lt_alldfields INTO ls_dfies
            WITH KEY fieldname = <lv_any>.
            IF sy-subrc <> 0.
*Specified field not found in table
              e_rc = 4.
              CONCATENATE 'FIELD' <lv_any> 'NOT FOUND IN TABLE' i_table
              INTO e_msg SEPARATED BY space.
              RETURN.
            ENDIF.
*Fieldname found; insert line into working dfies table
            INSERT ls_dfies INTO TABLE lt_dfields.
          ENDLOOP.
        ENDIF.

        CREATE DATA lr_dataref TYPE STANDARD TABLE OF (i_table).
        ASSIGN lr_dataref->* TO <lt_tab>.

        SELECT * FROM (i_table) INTO TABLE <lt_tab>
        WHERE (t_options).

        LOOP AT <lt_tab> ASSIGNING <ls_line>.

          FREE lt_stringtab.

          LOOP AT lt_dfields INTO ls_dfies.
            CLEAR: lv_string, lv_decrypt.
            ASSIGN COMPONENT ls_dfies-position
               OF STRUCTURE <ls_line> TO <lv_any>.
            lv_temp = <lv_any>. "force the type conversion

            "Zipped?
            IF ls_dfies-fieldname = 'COMPRESSION'.
              lv_zip = lv_temp.
            ENDIF.

            "special treatment for RAWSTRING fields
            IF ls_dfies-datatype = 'RSTR' AND i_decryption = 'X'.
              lv_rawstring = lv_temp.
              PERFORM decrypt_rawstring
                USING lv_zip lv_rawstring CHANGING lv_decrypt.
              e_string = lv_decrypt.
              lv_temp = lv_decrypt.
            ENDIF.

            lv_string = lv_temp.
            INSERT lv_string INTO TABLE lt_stringtab.
            "In case a string scan is done check if found
            IF i_scan_string IS NOT INITIAL.
              IF i_scan_field = ls_dfies-fieldname.
                IF lv_string CP i_scan_string.
                  CLEAR lv_skip.
                ELSE."string not found
                  lv_skip = 'X'.
                ENDIF.
              ENDIF.
```

```
            ENDIF.
          ENDLOOP.

          "In case scan had no hit -> no insert in result table
          IF lv_skip IS INITIAL.
            APPEND LINES OF lt_stringtab  TO e_stringtab.
          ENDIF.

          "Exit if a rowcount restriction was set
          IF i_rowcount > 0 AND sy-tabix GE i_rowcount.
            EXIT.
          ENDIF.

        ENDLOOP.

      CATCH cx_root INTO lr_error.
        e_rc = 4.
        e_msg = lr_error->get_text( ).
    ENDTRY.

  ELSE."read single line and field
    IF i_rawstring = 'X' AND i_decryption = 'X'.
      SELECT SINGLE (i_field) FROM (i_table)
         INTO lv_rawstring WHERE (t_options).
      CLEAR lv_zip.
      PERFORM decrypt_rawstring
         USING lv_zip lv_rawstring CHANGING lv_decrypt.
      e_string = lv_decrypt.
    "BEGIN NEW
    ELSEIF i_rawstring = 'X' AND i_decryption = ''.
      SELECT SINGLE (i_field) FROM (i_table)
       INTO lv_rawstring WHERE (t_options).
        e_string = lv_rawstring.
    "END NEW
    ELSE.
      SELECT SINGLE (i_field) FROM (i_table)
       INTO e_string WHERE (t_options).
    ENDIF.
  ENDIF.

ENDFUNCTION.

*&---------------------------------------------------------------------*
*&      Form  decrypt_rawstring
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM decrypt_rawstring USING lv_zip lv_rawstring CHANGING lv_decrypt.

  "Transform rawstring to text
  DATA: lv_converter TYPE REF TO cl_abap_conv_in_ce.
  DATA: lv_xstring   TYPE xstring.


*     unzip if compressed
  IF NOT lv_zip IS INITIAL.
    TRY.
        cl_abap_gzip=>decompress_binary(
                    EXPORTING
                         gzip_in = lv_rawstring
                    IMPORTING
                         raw_out = lv_xstring ).
      CATCH: cx_parameter_invalid_range cx_sy_buffer_overflow.
    ENDTRY.
  ELSE.
    lv_xstring = lv_rawstring.
  ENDIF.

*     read
  lv_converter = cl_abap_conv_in_ce=>create(
                    input       = lv_xstring
                    encoding    = 'UTF-8'
                    replacement = '?'
                    ignore_cerr = abap_true ).

  TRY.
      CALL METHOD lv_converter->read( IMPORTING data = lv_decrypt ).
    CATCH cx_sy_conversion_codepage.
```

```
*-- Should ignore errors in code conversions
    CATCH cx_sy_codepage_converter_init.
*-- Should ignore errors in code conversions
    CATCH cx_parameter_invalid_type.
    CATCH cx_parameter_invalid_range.

  ENDTRY.

ENDFORM.                    "decrypt_rawstring
```

11 Function Module Z_RFC_HCPR_CREATE - Creation of HCPRs

11.1 General Information

System	BI2
Function Module	Z_RFC_HCPR_CREATE, Creation of HCPRs
Function Pool	Z_DP
Remote	Yes
Last changed by	NMEYER
Last change (timestamp)	11/02/2019
Timestamp of documentation	11/08/2020 16:30:02

11.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
I_HCPRNM	TYPE	RSOHCPRNM	"		X	
I_DESCRIPTION	TYPE	SSTRING	"	X	X	
I_INFOAREA	TYPE	RSINFOAREA	"		X	

11.3 Export Parameter

Parameter		Associated Type	Pass	Short text
E_SUCCESS	TYPE	RS_BOOL	X	

11.4 Tables

Parameter		Associated Type	Opt.	Short text
T_PARTPROVIDER	LIKE	RSD_S_PROV		

11.5 Exceptions

Exception	Short text
RELEASE_1_0	

11.6 Source code Function module

```
FUNCTION Z_RFC_HCPR_CREATE.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_HCPRNM) TYPE  RSOHCPRNM DEFAULT ''
*"     VALUE(I_DESCRIPTION) TYPE  SSTRING DEFAULT ''
*"     VALUE(I_INFOAREA) TYPE  RSINFOAREA DEFAULT ''
*"  EXPORTING
*"     VALUE(E_SUCCESS) TYPE  RS_BOOL
*"  TABLES
*"      T_PARTPROVIDER STRUCTURE  RSD_S_PROV OPTIONAL
*"  EXCEPTIONS
*"      RELEASE_1_0
*"----------------------------------------------------------------------

  DATA: lt_partprovider TYPE TABLE OF rsinfoprov,
        ls_partprovider TYPE rsinfoprov,
        ls_temp         TYPE RSD_S_PROV.

  LOOP AT t_partprovider INTO ls_temp.
```

```
    ls_partprovider = ls_temp-infoprov.
    APPEND ls_partprovider TO lt_partprovider.
  ENDLOOP.


  CALL METHOD cl_rso_hcpr_metadata_api=>create_union
    EXPORTING
      i_hcprnm          = i_hcprnm
      i_description     = i_description
      i_infoarea        = i_infoarea
      i_t_part_provider = lt_partprovider
  IMPORTING
     "e_r_msg           = E_R_MSG
     e_success         = E_SUCCESS.

ENDFUNCTION.
```

12 Function Module Z_RFC_READ_REPORT - Read ABAP-Reports via RFC

12.1 General Information

System	BI2
Function Module	Z_RFC_READ_REPORT, Read ABAP-Reports via RFC
Function Pool	Z_DP
Remote	Yes
Last changed by	TSCHMIDT
Last change (timestamp)	24/08/2015
Timestamp of documentation	11/08/2020 16:30:05

12.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
PROGRAM	LIKE	SY-REPID			X	Report Name

12.3 Export Parameter

Parameter		Associated Type	Pass	Short text
SYSTEM	LIKE	SY-SYSID	X	Name of the SAP System
TRDIR	LIKE	TRDIR	X	Generated Table for View TRDIR

12.4 Tables

Parameter		Associated Type	Opt.	Short text
QTAB	LIKE	BAPITGB		Structure to transfer texts for BAPIs that read docu.

12.5 Exceptions

Exception	Short text
RELEASE_1_0	

12.6 Source code Function module

```
FUNCTION Z_RFC_READ_REPORT.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(PROGRAM) LIKE  SY-REPID
*"  EXPORTING
*"     VALUE(SYSTEM) LIKE  SY-SYSID
*"     VALUE(TRDIR) LIKE  TRDIR STRUCTURE  TRDIR
*"  TABLES
*"      QTAB STRUCTURE  BAPITGB
*"  EXCEPTIONS
*"      RELEASE_1_0
*"----------------------------------------------------------------------
```

```
REFRESH QTAB.
READ REPORT PROGRAM INTO QTAB.
SELECT SINGLE * FROM TRDIR WHERE NAME = PROGRAM.
SYSTEM = SY-SYSID.


ENDFUNCTION.
```

13 Function Module Z_RFC_TRANSLATION - Translation Steward

13.1 General Information

System	BI2
Function Module	Z_RFC_TRANSLATION, Translation Steward
Function Pool	Z_DP
Remote	Yes
Last changed by	MHARING
Last change (timestamp)	23/06/2020
Timestamp of documentation	11/08/2020 16:30:08

13.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
I_DELIMITER	LIKE	SONV-FLAG	'¦'		X	

13.3 Tables

Parameter		Associated Type	Opt.	Short text
T_VALUES	LIKE	TAB512		

13.4 Exceptions

Exception	Short text
RELEASE_1_2	

13.5 Source code Function module

```
FUNCTION Z_RFC_TRANSLATION.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(I_DELIMITER) LIKE  SONV-FLAG DEFAULT '¦'
*"  TABLES
*"      T_VALUES STRUCTURE  TAB512
*"  EXCEPTIONS
*"      RELEASE_1_2
*"----------------------------------------------------------------------

  FIELD-SYMBOLS: <fs_values> TYPE tab512,
                 <fieldname> TYPE string,
                 <target_struc> TYPE ANY,
                 <fieldvalue> TYPE ANY.

  DATA:
       ls_tab TYPE tab512,
       lt_fieldnames TYPE STANDARD TABLE OF string,
       lt_values TYPE STANDARD TABLE OF string,
       lr_target_struc TYPE REF TO data,
       lv_tabname TYPE tabname,
       lv_value TYPE string,
       lv_fieldname TYPE string.

  "get fields from first dataset
  READ TABLE t_values[] INTO ls_tab INDEX 1.
  IF sy-subrc <> 0.
    "empty table passed - not valid
    RETURN.
```

```
  ENDIF.


  LOOP AT t_values[] ASSIGNING <fs_values>.

    CLEAR: lv_value, lv_tabname.
    FREE: lt_fieldnames, lt_values.
    SPLIT <fs_values>  AT i_delimiter INTO TABLE lt_fieldnames.

    "Get table
    LOOP AT lt_fieldnames[] ASSIGNING <fieldname>.

      IF <fieldname>(5) EQ 'TABLE'.

        SPLIT <fieldname> AT '===' INTO TABLE lt_values.
        READ TABLE lt_values INTO lv_value INDEX 2.
        IF sy-subrc = 0.
          lv_tabname = lv_value.
          EXIT.
        ENDIF.

      ENDIF.
    ENDLOOP.

    "create table structure dynamically
    FREE lr_target_struc.
    CREATE DATA lr_target_struc TYPE (lv_tabname).
    ASSIGN lr_target_struc->* TO <target_struc>.

    "Write fields
    LOOP AT lt_fieldnames[] ASSIGNING <fieldname>.

      CLEAR: lv_value.
      FREE: lt_values.

      IF <fieldname>(5) EQ 'TABLE'.
        CONTINUE.
      ENDIF.

      SPLIT <fieldname> AT '===' INTO TABLE lt_values.
      READ TABLE lt_values INTO lv_fieldname INDEX 1.
      IF sy-subrc <> 0.
        CONTINUE.
      ENDIF.
      READ TABLE lt_values INTO lv_value INDEX 2.
      IF sy-subrc <> 0.
        CONTINUE.
      ENDIF.

      ASSIGN COMPONENT lv_fieldname OF STRUCTURE <target_struc>
       TO <fieldvalue>.
      IF sy-subrc <> 0.
        WRITE:/ 'FIELD ', <fieldname>, 'NOT FOUND'.
        CONTINUE.
      ENDIF.

      <fieldvalue> = lv_value.

    ENDLOOP.

    MODIFY (lv_tabname) FROM <target_struc>.

  ENDLOOP.

  CALL FUNCTION 'DB_COMMIT'.


ENDFUNCTION.
```

14 Function Module Z_RFC_USAGE_ANALYSIS - Where used analysis for BI Docu Performer

14.1 General Information

System	BI2

Function Module	Z_RFC_USAGE_ANALYSIS, Where used analysis for BI Docu Performer
Function Pool	Z_DP
Remote	Yes
Last changed by	NMEYER
Last change (timestamp)	12/10/2015
Timestamp of documentation	11/08/2020 16:30:12

14.2 Import Parameter

Parameter		Associated Type	Default	Opt.	Pass	Short text
I_VERSION	TYPE	RSOBJVERS	'A'		X	Object version
I_IOBJNM	TYPE	RSIOBJNM		X	X	InfoObject
I_ATTRINM	TYPE	RSATTRINM		X	X	Attribute (Display)
I_VALUE	TYPE	RSCHAVL		X	X	Value for Characteristic
I_COMPID	TYPE	RSZCOMPID		X	X	Name (ID) of a Reporting Component
I_UID	TYPE	SYSUUID_25		X	X	UUID in compressed form
I_EXTENDED	TYPE	SONV-FLAG		X	X	Extended check (incl. non-reusables)
I_LEVEL_MAX	TYPE	INT4	99	X	X	Max Level

14.3 Tables

Parameter		Associated Type	Opt.	Short text
T_CHAVL	LIKE	RSA_S_CHAVL		
T_DATA	LIKE	V_COMPDIR_COMPIC		
T_DATA2	LIKE	TAB512		
T_DATA_RANGE	LIKE	TAB512		

14.4 Exceptions

Exception	Short text
RELEASE_1_9	

14.5 Source code Function module

```
FUNCTION Z_RFC_USAGE_ANALYSIS.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_VERSION) TYPE  RSOBJVERS DEFAULT 'A'
*"     VALUE(I_IOBJNM) TYPE  RSIOBJNM OPTIONAL
*"     VALUE(I_ATTRINM) TYPE  RSATTRINM OPTIONAL
*"     VALUE(I_VALUE) TYPE  RSCHAVL OPTIONAL
*"     VALUE(I_COMPID) TYPE  RSZCOMPID OPTIONAL
*"     VALUE(I_UID) TYPE  SYSUUID_25 OPTIONAL
*"     VALUE(I_EXTENDED) TYPE  SONV-FLAG OPTIONAL
*"     VALUE(I_LEVEL_MAX) TYPE  INT4 DEFAULT 99
*"  TABLES
*"      T_CHAVL STRUCTURE  RSA_S_CHAVL
*"      T_DATA STRUCTURE  V_COMPDIR_COMPIC
*"      T_DATA2 STRUCTURE  TAB512
*"      T_DATA_RANGE STRUCTURE  TAB512
*"  EXCEPTIONS
*"      RELEASE_1_9
*"----------------------------------------------------------------------


* A] Elemente in Query-Selektionen  -> RSZRANGE
* B] Berechnete Kennzahlen -> RSZCALC
*---------------------------------------------------------------------

  TYPES:
    BEGIN OF ty_uid_deftp,
      compuid TYPE sysuuid_25,
      deftp   TYPE rszdeftp,
    END OF ty_uid_deftp.

  DATA:
```

```
    l_subrc        TYPE rs_bool,
    l_iobjnm       TYPE rsiobjnm,
    l_uid          TYPE sysuuid_25,
    l_iobjtp       TYPE rsiobjtp,
    l_s_uid_deftp  TYPE ty_uid_deftp,
    l_t_data       TYPE ty_t_data_scan,
    l_t_data_range TYPE ty_t_data_range,
    l_s_data       TYPE v_compdir_compic,"Struktur für Export Parameter
    l_s_data2      TYPE char512,         "Struktur für Export Parameter
    l_s_data_range TYPE char512.         "Struktur für Export Parameter

  FIELD-SYMBOLS:
    <s_data>       TYPE ty_s_data_scan,
    <s_data_range> TYPE ty_s_data_range,
    <fs_chavl>     TYPE rsa_s_chavl.

  LOOP AT t_chavl ASSIGNING <fs_chavl>.
    ls_chavl-sign   = 'I'.
    ls_chavl-option = 'EQ'.
    ls_chavl-low    = <fs_chavl>-chavl.
    APPEND ls_chavl TO gt_chavl.
  ENDLOOP.

  "Globalen Parameter setzen
  p_level_max = i_level_max.

  "Wiederverwendbare Elemente puffern
  SELECT * FROM rszcompdir INTO TABLE g_t_rszcompdir
     WHERE objvers = 'A'.

  IF i_iobjnm IS NOT INITIAL.
    "---------------------------------------------------------------------
    " Where-used analysis for InfoObjects
    "---------------------------------------------------------------------

    "Check if a display attribute is entered:
    IF i_attrinm IS NOT INITIAL.
      PERFORM scan_attr_dis USING i_iobjnm i_attrinm i_version
                            CHANGING l_t_data.
    ELSE.
      SELECT SINGLE iobjtp FROM rsdiobj INTO l_iobjtp
         WHERE iobjnm = i_iobjnm.
      IF l_iobjtp = 'KYF'.
        PERFORM scan_kyfnm USING i_iobjnm i_version CHANGING l_t_data.
      ELSEIF i_value IS NOT INITIAL OR gt_chavl IS NOT INITIAL.
        PERFORM scan_value_sel
            USING i_value i_iobjnm i_version gt_chavl
            CHANGING l_t_data l_t_data_range.
      ELSE.
*       l_seltp = rsd_c_metaiobj-charact.
        PERFORM scan_char USING i_iobjnm i_version CHANGING l_t_data.
      ENDIF.
    ENDIF.

  ELSEIF i_compid IS NOT INITIAL.
    "---------------------------------------------------------------------
    " Analysis for Reusable Query Elements
    "---------------------------------------------------------------------

    "Get type and uid from view V_COMPDIR_ELTDIR
    SELECT SINGLE compuid deftp FROM v_compdir_eltdir
           INTO l_s_uid_deftp WHERE compid = i_compid.
    IF sy-subrc = 0.
      l_uid   = l_s_uid_deftp-compuid.
      CLEAR l_iobjnm.
*     CASE l_deftp.
      CASE l_s_uid_deftp-deftp.
        WHEN 'VAR'.
          "Get iobjnm from view rszglobv
          SELECT SINGLE iobjnm FROM rszglobv
                 INTO l_iobjnm WHERE vnam = i_compid.
      ENDCASE.
      PERFORM scan_compid USING i_compid i_version l_iobjnm l_uid
                          CHANGING l_t_data.
    ENDIF.

  ELSEIF i_uid IS NOT INITIAL.
    "---------------------------------------------------------------------
```

```
    " Analysis for UID
    "---------------------------------------------------------------------
    IF i_extended IS INITIAL.
      SELECT SINGLE deftp FROM v_compdir_eltdir
        INTO CORRESPONDING FIELDS OF l_s_uid_deftp
        WHERE compuid = i_uid.
    ELSE.
      SELECT SINGLE deftp FROM rszeltdir
      INTO CORRESPONDING FIELDS OF l_s_uid_deftp
      WHERE eltuid = i_uid.
    ENDIF.
    IF sy-subrc = 0.
      l_uid   = i_uid.
      PERFORM scan_compid USING i_compid i_version l_iobjnm l_uid
                          CHANGING l_t_data.
    ENDIF.

  ENDIF.

  "---------------------------------------------------------------------
  " Exportparameter füllen
  "---------------------------------------------------------------------
  "Tabelle T_DATA + T_DATA2
  LOOP AT l_t_data ASSIGNING <s_data>.
    AT FIRST.
      "Header Zeile für Exportparameter T_DATA2 erzeugen
      l_s_data2 = 'COMPUID;LEVEL;DEFT;COMPID;COMPUID_PARENT'.
      APPEND l_s_data2 TO t_data2[].
    ENDAT.

    "Parameter T_DATA
    MOVE-CORRESPONDING <s_data> TO l_s_data.
    APPEND l_s_data TO t_data[].

    "Parameter T_DATA2
    CLEAR l_s_data2.
    CONCATENATE
        <s_data>-compuid
        <s_data>-level
        <s_data>-deftp
        <s_data>-compid
        <s_data>-compuid_parent
      INTO l_s_data2 SEPARATED BY ';'.
    APPEND l_s_data2 TO t_data2[].
  ENDLOOP.

  "Doppelte Einträge löschen
  SORT t_data[] BY compuid.
  DELETE ADJACENT DUPLICATES FROM t_data[].

  "Tabelle T_DATA_RANGE
  LOOP AT l_t_data_range ASSIGNING <s_data_range>.
    AT FIRST.
      "Header Zeile für Exportparameter T_DATA_RANGE erzeugen
      l_s_data_range =
        'COMPUID;IOBJNM;SIGN;OPT;LOW;LOWFLAG;HIGH;HIGHFLAG;HIENM'.
      APPEND l_s_data_range TO t_data_range[].
    ENDAT.

    READ TABLE l_t_data ASSIGNING <s_data>
       WITH KEY compuid_parent = <s_data_range>-compuid.
    IF sy-subrc = 0.
      <s_data_range>-compuid = <s_data>-compuid.
    ENDIF.

    CLEAR l_s_data_range.
    CONCATENATE
        <s_data_range>-compuid
        <s_data_range>-iobjnm
        <s_data_range>-sign
        <s_data_range>-opt
        <s_data_range>-low
        <s_data_range>-lowflag
        <s_data_range>-high
        <s_data_range>-highflag
        <s_data_range>-hienm
      INTO l_s_data_range SEPARATED BY ';'.
    APPEND l_s_data_range TO t_data_range[].
```

```
  ENDLOOP.

ENDFUNCTION.


*&---------------------------------------------------------------------*
*&      Form  scan_attr_dis
*&---------------------------------------------------------------------*
FORM scan_attr_dis
  USING
    i_iobjnm
    i_attrinm
    i_version
  CHANGING
    c_t_data.

  DATA:
    l_t_eltuid  TYPE ty_t_eltuid,
    l_eltuid    TYPE sysuuid_25,
    l_s_data    TYPE ty_s_data_scan.

  FIELD-SYMBOLS:
    <s_data>    TYPE rsz_x_eltdir,
    <s_eltuid>  TYPE ty_s_eltuid.

  FREE l_t_eltuid.

  SELECT DISTINCT eltuid FROM rszeltattr INTO TABLE l_t_eltuid
    WHERE objvers = i_version
      AND iobjnm  = i_iobjnm
      AND attrinm = i_attrinm.

  "Elemente anfügen und an den Elementen hochhangeln
  LOOP AT l_t_eltuid ASSIGNING <s_eltuid>.
    l_eltuid = <s_eltuid>-eltuid.

    CLEAR l_s_data.
    l_s_data-compuid_parent = l_eltuid.

    PERFORM scan_query_elem_parent
      USING
        i_iobjnm
        l_eltuid
        i_version
      CHANGING
        c_t_data
        l_s_data.
  ENDLOOP.

ENDFORM.                    "scan_attr_dis


*&---------------------------------------------------------------------*
*&      Form  scan_kyfnm
*&---------------------------------------------------------------------*
FORM scan_kyfnm
  USING
    i_iobjnm
    i_version
  CHANGING
    c_t_data.

  DATA:
    l_t_eltuid  TYPE ty_t_eltuid,
    l_eltuid    TYPE sysuuid_25,
    l_s_data    TYPE ty_s_data_scan.

  FIELD-SYMBOLS:
    <s_data>    TYPE rsz_x_eltdir,
    <s_eltuid>  TYPE ty_s_eltuid.

  "---------------------------------------------------------------
  " A] Kennzahl in Tabelle RSZRANGE suchen...
  "    (Eingeschränkte Kennzahl oder direkt in Query)
  "---------------------------------------------------------------
  FREE l_t_eltuid.

  SELECT DISTINCT eltuid FROM rszrange INTO TABLE l_t_eltuid
```

```
    WHERE objvers = i_version
      AND iobjnm  = rsd_c_metaiobj-keyfigure
      AND seltp   = rzd1_c_seltp-keyfig
      AND low     = i_iobjnm.

  "Elemente anfügen und an den Elementen hochhangeln
  LOOP AT l_t_eltuid ASSIGNING <s_eltuid>.
    l_eltuid = <s_eltuid>-eltuid.

    CLEAR l_s_data.
    l_s_data-compuid_parent = l_eltuid.

    PERFORM scan_query_elem_parent
      USING
        i_iobjnm
        l_eltuid
        i_version
      CHANGING
        c_t_data
        l_s_data.
  ENDLOOP.

  "--------------------------------------------------------------------
  " B] Kennzahl in Tabelle RSZCALC suchen...
  "    (Formel und berechnete Kennzahl)
  "--------------------------------------------------------------------
  FREE l_t_eltuid.
  SELECT DISTINCT eltuid FROM rszcalc INTO TABLE l_t_eltuid
    WHERE ( objvers = i_version                    )
      AND ( oper1   = i_iobjnm OR oper2 = i_iobjnm ).

  "Elemente anfügen und an den Elementen hochhangeln
  LOOP AT l_t_eltuid ASSIGNING <s_eltuid>.
    l_eltuid = <s_eltuid>-eltuid.

    CLEAR l_s_data.
    l_s_data-compuid_parent = l_eltuid.

    PERFORM scan_query_elem_parent
      USING
        i_iobjnm
        l_eltuid
        i_version
      CHANGING
        c_t_data
        l_s_data.
  ENDLOOP.

ENDFORM.                    "scan_kyfnm


*&---------------------------------------------------------------------*
*&      Form  scan_value_sel
*&---------------------------------------------------------------------*
*       Search for characteristics with special selection
*----------------------------------------------------------------------*
FORM scan_value_sel
  USING
    i_value
    i_iobjnm
    i_version
    lt_chavl LIKE gt_chavl
  CHANGING
    c_t_data
    c_t_data_range TYPE ty_t_data_range.

  DATA:
    l_value        TYPE rschavl,
    l_eltuid       TYPE sysuuid_25,
    l_s_data       TYPE ty_s_data_scan,
    l_t_data_range TYPE ty_t_data_range.

  FIELD-SYMBOLS:
    <s_data_range> TYPE ty_s_data_range.

  FREE l_t_data_range.

  IF i_value CS '*' OR i_value CS '%'.
```

```
    l_value = i_value.
    REPLACE ALL OCCURRENCES OF '*' IN l_value WITH '%'.
    SELECT DISTINCT
        eltuid
        iobjnm
        sign
        opt
        low
        lowflag
        high
        highflag
        hienm
      FROM rszrange INTO CORRESPONDING FIELDS OF TABLE l_t_data_range
      WHERE objvers = i_version
        AND iobjnm  LIKE i_iobjnm "1_8; support Nav-Attributes
         AND ( ( opt = 'EQ' AND low LIKE l_value AND lowflag = 1 )
           OR ( opt = 'BT' AND low <= l_value AND lowflag = 1
             AND high >= l_value AND highflag = 1 ) ).
  ELSEIF i_value IS NOT INITIAL.
    SELECT DISTINCT
        eltuid
        iobjnm
        sign
        opt
        low
        lowflag
        high
        highflag
        hienm
      FROM rszrange INTO CORRESPONDING FIELDS OF TABLE l_t_data_range
      WHERE objvers = i_version
        AND iobjnm  LIKE i_iobjnm"1_8; support Nav-Attributes
        AND ( ( opt = 'EQ' AND low = i_value )
         OR ( opt = 'BT' AND low <= i_value AND lowflag = 1
               AND high >= i_value  AND highflag = 1 ) ).
  ELSEIF lt_chavl IS NOT INITIAL. "Several values are passed (in table)
    SELECT DISTINCT
        eltuid
        iobjnm
        sign
        opt
        low
        lowflag
        high
        highflag
        hienm
      FROM rszrange INTO CORRESPONDING FIELDS OF TABLE l_t_data_range
      WHERE objvers =  i_version
        AND iobjnm  LIKE  i_iobjnm "1_8; support Nav-Attributes
        AND low     IN lt_chavl.
  ENDIF.

  "Elemente anfügen und an den Elementen hochhangeln
  LOOP AT l_t_data_range ASSIGNING <s_data_range>.
    l_eltuid = <s_data_range>-eltuid.

    CLEAR l_s_data.
    l_s_data-compuid        = l_eltuid.
    l_s_data-compuid_parent = l_eltuid.

    <s_data_range>-compuid = l_eltuid.
    APPEND <s_data_range> TO c_t_data_range.

    PERFORM scan_query_elem_parent
      USING
        i_iobjnm
        l_eltuid
        i_version
      CHANGING
        c_t_data
        l_s_data.
  ENDLOOP.

ENDFORM.                    "scan_value_sel

*&---------------------------------------------------------------------*
*&      Form  scan_char
*&---------------------------------------------------------------------*
```

```
*---------------------------------------------------------------------*
* A] Merkmale in Queries             -> RSZELTDIR
* B] Merkmale in Queries             -> RSZSELECT
* C] Merkmale in Ausnahmeaggregation -> RSZCALC
* D] Merkmale in Variablen           -> RSZGLOBV
* E] Merkmale als Attribut           -> RSZELTATTR
*---------------------------------------------------------------------*
FORM scan_char
  USING
    i_iobjnm
    i_version
  CHANGING
    c_t_data.

  DATA:
    l_t_eltuid     TYPE ty_t_eltuid,
    l_eltuid       TYPE sysuuid_25,
    l_s_rszeltxref TYPE ty_s_rszeltxref,
    l_s_rszelttxt  TYPE rszelttxt,
    l_subrc        TYPE rs_bool,
    l_srch_attr    TYPE string,
    l_s_data       TYPE ty_s_data_scan.

  FIELD-SYMBOLS:
    <s_data>       TYPE ty_s_data_scan,
    <s_eltuid>     TYPE ty_s_eltuid,
    <s_rszcompdir> TYPE rszcompdir.

  CONCATENATE '%__' i_iobjnm INTO l_srch_attr.

  "---------------------------------------------------------------------
  " A] RSZELTDIR
  "---------------------------------------------------------------------
  FREE l_t_eltuid.
  SELECT DISTINCT eltuid FROM rszeltdir
    INTO CORRESPONDING FIELDS OF TABLE l_t_eltuid
    WHERE ( objvers     = i_version                                    )
      AND ( defaulthint = i_iobjnm OR defaulthint LIKE l_srch_attr ).

  "Elemente anfügen und an den Elementen hochhangeln
  LOOP AT l_t_eltuid ASSIGNING <s_eltuid>.
    l_eltuid = <s_eltuid>-eltuid.

    CLEAR l_s_data.
    l_s_data-compuid_parent = l_eltuid.

    PERFORM scan_query_elem_parent
      USING
        i_iobjnm
        l_eltuid
        i_version
      CHANGING
        c_t_data
        l_s_data.
  ENDLOOP.

  "---------------------------------------------------------------------
  " B] RSZSELECT
  "---------------------------------------------------------------------
  FREE l_t_eltuid.
  SELECT DISTINCT eltuid FROM rszselect
    INTO CORRESPONDING FIELDS OF TABLE l_t_eltuid
    WHERE objvers = i_version
      AND ( ( iobjnm = i_iobjnm OR coniobjnm = i_iobjnm )
       OR ( iobjnm LIKE l_srch_attr OR coniobjnm LIKE l_srch_attr ) ).

  "Elemente anfügen und an den Elementen hochhangeln
  LOOP AT l_t_eltuid ASSIGNING <s_eltuid>.
    l_eltuid = <s_eltuid>-eltuid.

    CLEAR l_s_data.
    l_s_data-compuid_parent = l_eltuid.

    PERFORM scan_query_elem_parent
      USING
        i_iobjnm
        l_eltuid
        i_version
```

```
    CHANGING
      c_t_data
      l_s_data.
ENDLOOP.

"---------------------------------------------------------------------
" C] RSZCALC
"---------------------------------------------------------------------
FREE l_t_eltuid.
SELECT DISTINCT eltuid FROM rszcalc INTO TABLE l_t_eltuid
  WHERE objvers = i_version
    AND ( aggrcha = i_iobjnm
     OR   aggrcha LIKE l_srch_attr ).

"Elemente anfügen und an den Elementen hochhangeln
LOOP AT l_t_eltuid ASSIGNING <s_eltuid>.
  l_eltuid = <s_eltuid>-eltuid.

  CLEAR l_s_data.
  l_s_data-compuid_parent = l_eltuid.

  PERFORM scan_query_elem_parent
    USING
      i_iobjnm
      l_eltuid
      i_version
    CHANGING
      c_t_data
      l_s_data.
ENDLOOP.

"---------------------------------------------------------------------
" D] RSZGLOBV
"---------------------------------------------------------------------
FREE l_t_eltuid.
SELECT DISTINCT varuniid FROM rszglobv INTO TABLE l_t_eltuid
  WHERE objvers = i_version
    AND ( iobjnm = i_iobjnm
     OR   iobjnm LIKE l_srch_attr ).

"Elemente anfügen und an den Elementen hochhangeln
LOOP AT l_t_eltuid ASSIGNING <s_eltuid>.
  l_eltuid = <s_eltuid>-eltuid.

  CLEAR l_s_data.
  l_s_data-compuid_parent = l_eltuid.

  PERFORM scan_query_elem_parent
    USING
      i_iobjnm
      l_eltuid
      i_version
    CHANGING
      c_t_data
      l_s_data.
ENDLOOP.

"---------------------------------------------------------------------
" E] RSZELTATTR
"---------------------------------------------------------------------
"In case of 7.x Queries: Attributes are yet detected by selection
"on table RSZELTDIR, but in case of 3.x Queries they have to be
"detected by reading table RSZELTATTR
FREE l_t_eltuid.
SELECT DISTINCT eltuid FROM rszeltattr INTO TABLE l_t_eltuid
  WHERE objvers = i_version
    AND attrinm = i_iobjnm.

"Elemente anfügen und an den Elementen hochhangeln
LOOP AT l_t_eltuid ASSIGNING <s_eltuid>.
  l_eltuid = <s_eltuid>-eltuid.

  CLEAR l_s_data.
  l_s_data-compuid_parent = l_eltuid.

  PERFORM scan_query_elem_parent
    USING
      i_iobjnm
```

```
        l_eltuid
        i_version
      CHANGING
        c_t_data
        l_s_data.
  ENDLOOP.

ENDFORM.                    "scan_char


*&---------------------------------------------------------------------*
*&      Form  scan_compid
*&---------------------------------------------------------------------*
FORM scan_compid
  USING
    i_compid
    i_version
    i_iobjnm
    i_uid
  CHANGING
    c_t_data.

  DATA:
    l_eltuid       TYPE sysuuid_25,
    l_s_rszeltxref TYPE ty_s_rszeltxref,
    l_t_rszeltxref TYPE TABLE OF ty_s_rszeltxref,
    l_s_data       TYPE ty_s_data_scan.

  SELECT DISTINCT seltuid teltuid laytp FROM rszeltxref
    INTO TABLE l_t_rszeltxref
    WHERE objvers = i_version
      AND teltuid = i_uid.

  "Elemente anfügen und an den Elementen hochhangeln
  LOOP AT l_t_rszeltxref INTO l_s_rszeltxref.
    l_eltuid = l_s_rszeltxref-seltuid.

    CLEAR l_s_data.
    l_s_data-compuid_parent = l_eltuid.

    PERFORM scan_query_elem_parent
      USING
        i_iobjnm
        l_eltuid
        i_version
      CHANGING
        c_t_data
        l_s_data.

  ENDLOOP.

ENDFORM.                    "scan_compid


*&---------------------------------------------------------------------*
*&      Form  SCAN_QUERY_ELEM_PARENT
*&---------------------------------------------------------------------*
FORM scan_query_elem_parent
  USING
    i_iobjnm
    i_teltuid
    i_version
  CHANGING
    c_t_data         TYPE ty_t_data_scan
    c_s_data_parent  TYPE ty_s_data_scan.

  DATA:
    l_s_rszeltxref   TYPE ty_s_rszeltxref,
    l_t_rszeltxref   TYPE ty_t_rszeltxref,
    l_level          TYPE c LENGTH 4,
    l_compuid        TYPE sysuuid_25.

  "Handelt es sich um ein wiederverw. Query Element? => hinzufügen
  PERFORM add_record
    USING
      i_iobjnm
      i_teltuid
      i_version
```

```
    CHANGING
      c_t_data
      c_s_data_parent.

  l_level   = c_s_data_parent-level.
  l_compuid = c_s_data_parent-compuid.

  "Übergeordnete Elemete ermitteln (SELTUID)
  SELECT seltuid teltuid laytp FROM rszeltxref
    INTO TABLE l_t_rszeltxref
    WHERE objvers = 'A'
      AND teltuid = i_teltuid.

  LOOP AT l_t_rszeltxref INTO l_s_rszeltxref.

    c_s_data_parent-level   = l_level.
    c_s_data_parent-compuid = l_compuid.

    "Rekursion beenden?
    CHECK l_level < p_level_max.

    "Eine Stufe hoch (die oberste Ebene entspricht der Query)
    PERFORM scan_query_elem_parent
      USING
        i_iobjnm
        l_s_rszeltxref-seltuid
        i_version
      CHANGING
        c_t_data
        c_s_data_parent.

  ENDLOOP.

ENDFORM.                    "SCAN_QUERY_ELEM_PARENT


*&---------------------------------------------------------------------*
*&      Form  add_record
*&---------------------------------------------------------------------*
FORM add_record
  USING
    i_iobjnm        TYPE rsiobjnm
    i_eltuid        TYPE sysuuid_25
    i_version       TYPE rsobjvers
  CHANGING
    c_t_data        TYPE ty_t_data_scan
    c_s_data_parent TYPE ty_s_data_scan.

  DATA:
    l_s_data        TYPE ty_s_data_scan,
    l_s_rszglobv    TYPE rszglobv,
    l_level         TYPE c LENGTH 4.

  FIELD-SYMBOLS:
    <s_rszcompdir> TYPE rszcompdir.

  "---------------------------------------------------------------------
  " Wiederverwendbares Element?
  "---------------------------------------------------------------------
  READ TABLE g_t_rszcompdir ASSIGNING <s_rszcompdir>
   WITH TABLE KEY compuid = i_eltuid.
  IF sy-subrc = 0.

    CLEAR l_s_data.
    MOVE <s_rszcompdir>-compuid  TO l_s_data-compuid.
    MOVE <s_rszcompdir>-compid   TO l_s_data-compid.
    MOVE <s_rszcompdir>-version  TO l_s_data-version.
    MOVE <s_rszcompdir>-tstpnm   TO l_s_data-tstpnm.
    MOVE <s_rszcompdir>-timestmp TO l_s_data-timestmp.

    l_level = c_s_data_parent-level + '0001'.
    MOVE l_level                 TO l_s_data-level.
    MOVE c_s_data_parent-compuid TO l_s_data-compuid_parent.

    "Datensatz schon vorhanden?
    READ TABLE c_t_data TRANSPORTING NO FIELDS
      WITH KEY
        compuid        = <s_rszcompdir>-compuid
```

```
        compuid_parent = l_s_data-compuid_parent.

    IF sy-subrc NE 0.

*    IF l_s_data-compuid NE l_s_data-compuid_parent.

      IF l_s_data-compuid EQ l_s_data-compuid_parent.
        CLEAR l_s_data-compuid_parent.
      ENDIF.

      "Zusätzl. Infos aus RSZCOMPIC
      SELECT SINGLE infocube FROM rszcompic INTO l_s_data-infocube
        WHERE compuid = i_eltuid
          AND objvers = i_version.

      "Zusätzl. Infos lesen aus RSZELTDIR
      SELECT SINGLE deftp FROM rszeltdir INTO (l_s_data-deftp)
        WHERE eltuid  = i_eltuid
          AND objvers = i_version.

      "Zusätzl. Infos aus V_CMP_JOIN
      SELECT SINGLE deftp FROM v_cmp_join INTO (l_s_data-deftp)
        WHERE compuid = i_eltuid
          AND objvers = i_version.

      "Text lesen aus RSZELTTXT
      SELECT SINGLE txtlg FROM rszelttxt INTO l_s_data-txtlg
        WHERE eltuid  = l_s_data-compuid
          AND objvers = i_version
          AND langu   = sy-langu.

      INSERT l_s_data INTO TABLE c_t_data.

*     ENDIF.

      "Letzten Satz für Level und Vorgänger merken
      c_s_data_parent = l_s_data.

    ENDIF.

    RETURN.

  ENDIF.

*  "---------------------------------------------------------------------
*  " Variable?
*  "---------------------------------------------------------------------
  SELECT SINGLE * FROM rszglobv INTO l_s_rszglobv
    WHERE varuniid = i_eltuid
      AND objvers  = 'A'.

  IF sy-subrc = 0.
    l_s_data-compuid = l_s_rszglobv-varuniid.
    l_s_data-compid  = l_s_rszglobv-vnam.
    INSERT l_s_data INTO TABLE c_t_data.
  ENDIF.

ENDFORM.                    " ADD_RECORD
```