PUBLIC

How-To: Extend the Business Partner – Node Extension (Reuse Option)

Applicable Releases:

all

Version 2.0

October 2023

Document History

Document Version	Description
1.0	First official release of this guide (July 2021)
2.0	Layout update (October 2023)

1 Business Scenario

SAP Master Data Governance (MDG) provides business processes to find, create, change, and mark master data for deletion. It supports the governance of master data in a central hub and the distribution to connected operational and business intelligence systems.

The processes are workflow-driven and can include several approval and revision phases, and the collaboration of all users participating in the master data maintenance.

MDG offers change request (CR)-based processing of master data with integrated workflow, staging, approval, activation, and distribution.

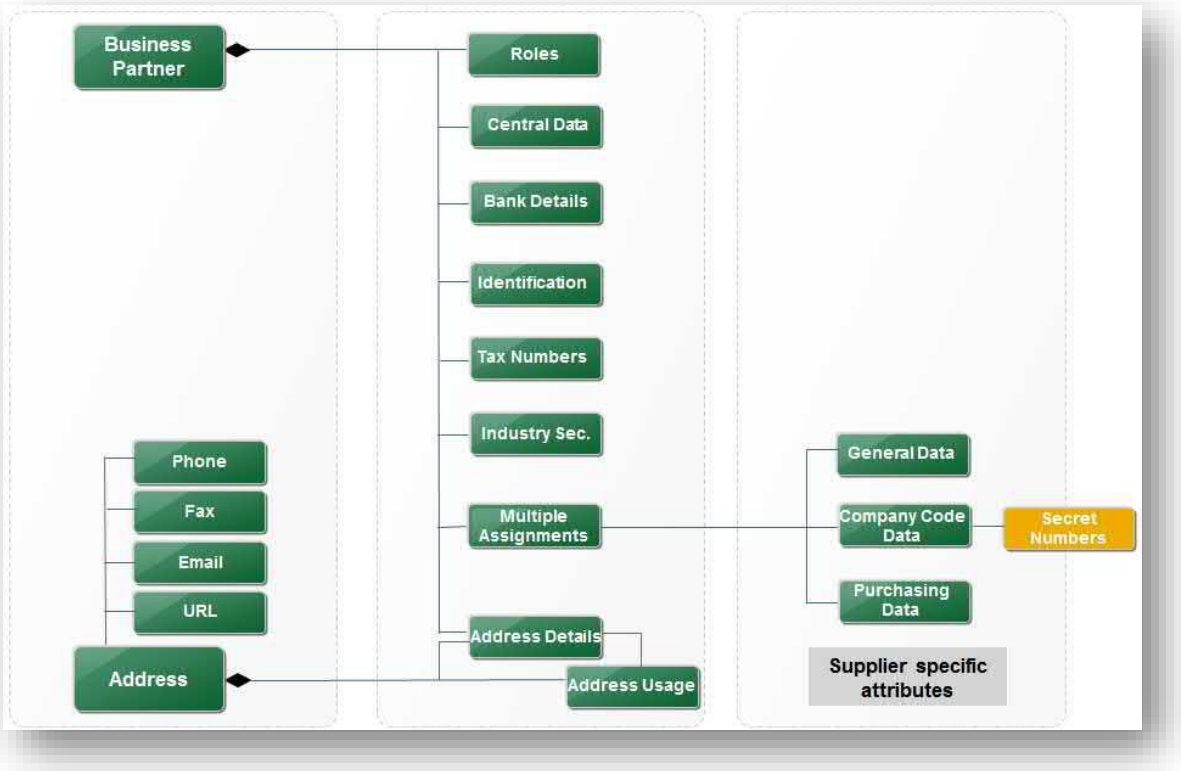
If your domain-specific solution does not fully meet requirements, you can customize and extend it. You can use this guide to extend the data model for supplier governance (MDG-S) or for customer governance (MDG-C) by creating a new node, using the reuse entity type.

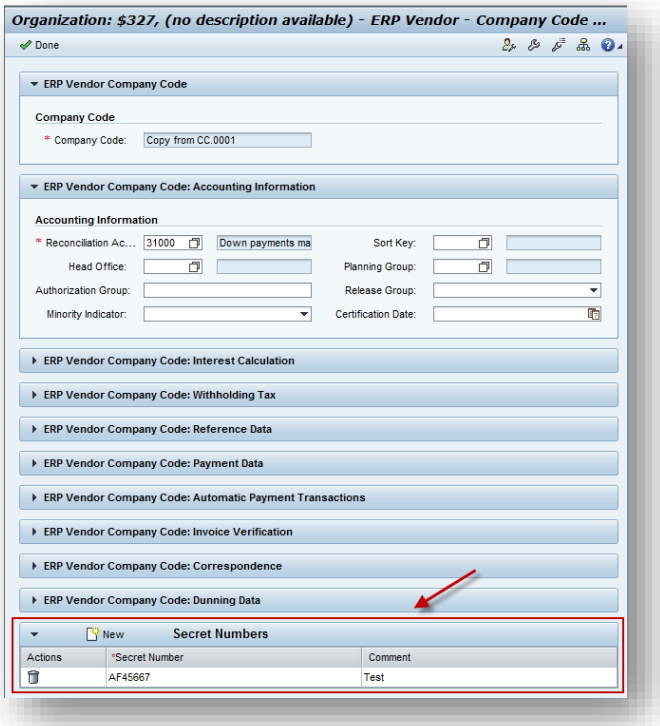
This document explains how to add a new node to Customer Governance (MDG-C) or Supplier Governance(MDG-S) using a reuse entity. The given scenario is complex. It is recommended you study the following How-To Guides before working with this one:

Overview of MDG-BP/C/S	This guide provides you with foundation knowledge about business partner, customer and supplier data and its related governance solutions business partner governance (MDG-BP), customer governance (MDG-C) and supplier governance (MDG-S).
Create a Custom Handler Class	This guide shows how to extend the MDG Business Partner / Customer / Supplier by creating and registering a custom handler class. This guide also includes an example that describes how to change the behavior of the existing MDG-S UI so that it always creates the ERP vendor record.
Create or Redefine a UI Feeder Class	This guide describes how to extend the MDG Business Partner / Customer / Supplier solution by either creating or redefining a custom feeder class for the user interface. It explains how this feeder class is added to an existing User Interface Building Block (UIBB).

2 Scenario

In addition to managing company code data within supplier governance, you want to manage a secret number and a comment. During processing of the change request, you want the additional attributes to bestored in the MDG staging area. After activation, you want the attributes to be stored in a custom Z-table.





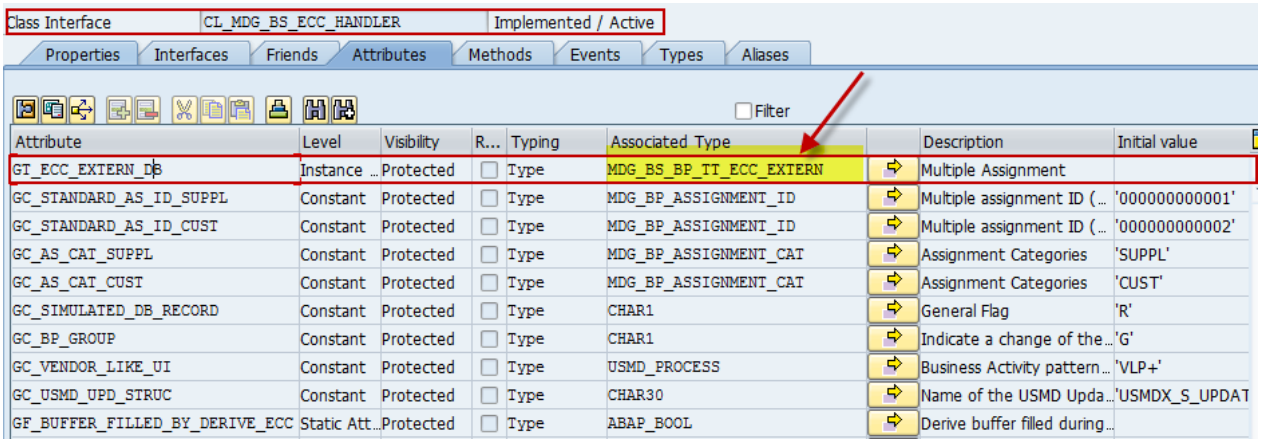
3 Technical Background

Before a new table-like entity type can be handled properly in MDG, you must extend the following:

- The MDG data model
- The structures used by a custom handler class
- The Business Object Layer (BOL) / Generic Integration Layer (genIL) model
- The User Interface.

To handle an additional table-like entity type in the custom handler class, you must extend structure `MDG_BS_BP_TT_ECC_EXTERN`.

CL_MDG_BS_ECC_HANDLER



In structure **MDG_BS_BP_TT_ECC_EXTERN** identify and extend the relevant substructure. If you want to adddata below the company element, you must extend the structure as shown below.

VMDS_EI_VMD_COMPANY.MDG_BS_BP_TT_ECC_EXTERN

- **MDG_BS_BP_TT_MLT_AS_SUPPL**
  - **VMDS_EI_EXTERN**
    - **VMDS_EI_VMD_COMPANY**

The new substructure must adhere to a certain format. It is a good practice to follow the pattern used for existing structures which are also located below **VMDS_EI_VMD_COMPANY** such as **DUNNING** or **WTAX_TYPE**.

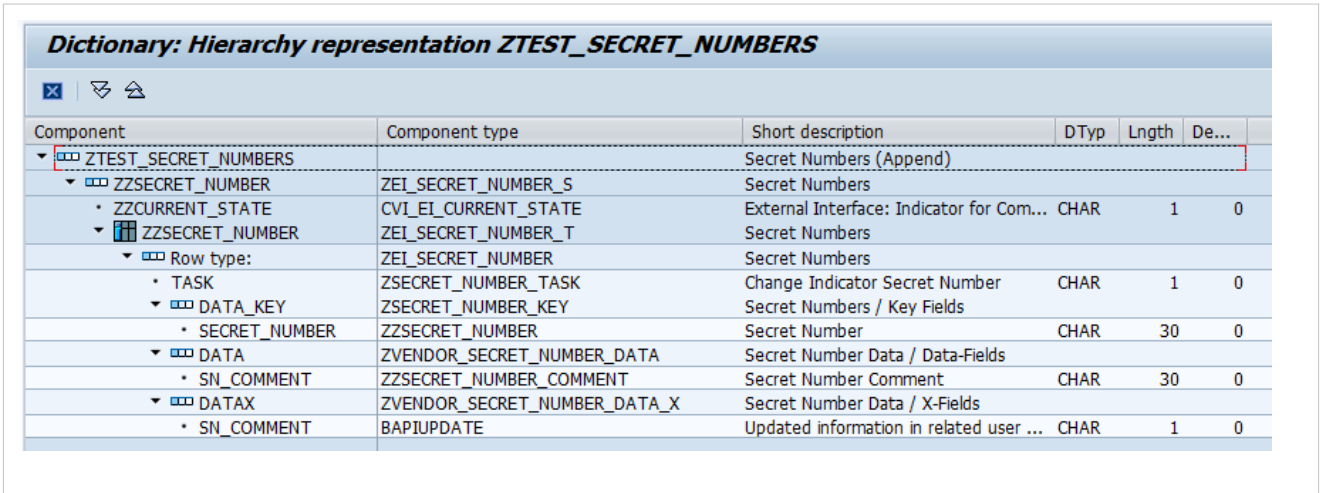
4 Step by Step Implementation Guide

The following steps provide details how-to extend the MDG-S model by a new table-like entity type.

4.1 Extend structure VMDS_EI_VMD_COMPANY

As a starting point you will create the necessary DDIC object to create the structure ZTEST_SECRET_NUMBERS

ZTEST_SECRET_NUMBERS



Dictionary: Hierarchy representation ZTEST_SECRET_NUMBERS

Component	Component type	Short description	DTyp	Lngth	De...
ZTEST_SECRET_NUMBERS		Secret Numbers (Append)			
ZZSECRET_NUMBER	ZEI_SECRET_NUMBER_S	Secret Numbers			
ZZCURRENT_STATE	CVI_EI_CURRENT_STATE	External Interface: Indicator for Com...	CHAR	1	0
ZZSECRET_NUMBER	ZEI_SECRET_NUMBER_T	Secret Numbers			
Row type:	ZEI_SECRET_NUMBER	Secret Numbers			
TASK	ZSECRET_NUMBER_TASK	Change Indicator Secret Number	CHAR	1	0
DATA_KEY	ZSECRET_NUMBER_KEY	Secret Numbers / Key Fields			
SECRET_NUMBER	ZZSECRET_NUMBER	Secret Number	CHAR	30	0
DATA	ZVENDOR_SECRET_NUMBER_DATA	Secret Number Data / Data-Fields			
SN_COMMENT	ZZSECRET_NUMBER_COMMENT	Secret Number Comment	CHAR	30	0
DATAX	ZVENDOR_SECRET_NUMBER_DATA_X	Secret Number Data / X-Fields			
SN_COMMENT	BAPIUPDATE	Updated information in related user ...	CHAR	1	0

For details of the individual objects in the data dictionary, refer to the Appendix section Data Dictionary Objects for Extension of VMDS_EI_VMD_COMPANY.

Table Types

Name	Description
ZEI_SECRET_NUMBER_T	Vendors: Secret Numbers

Structures

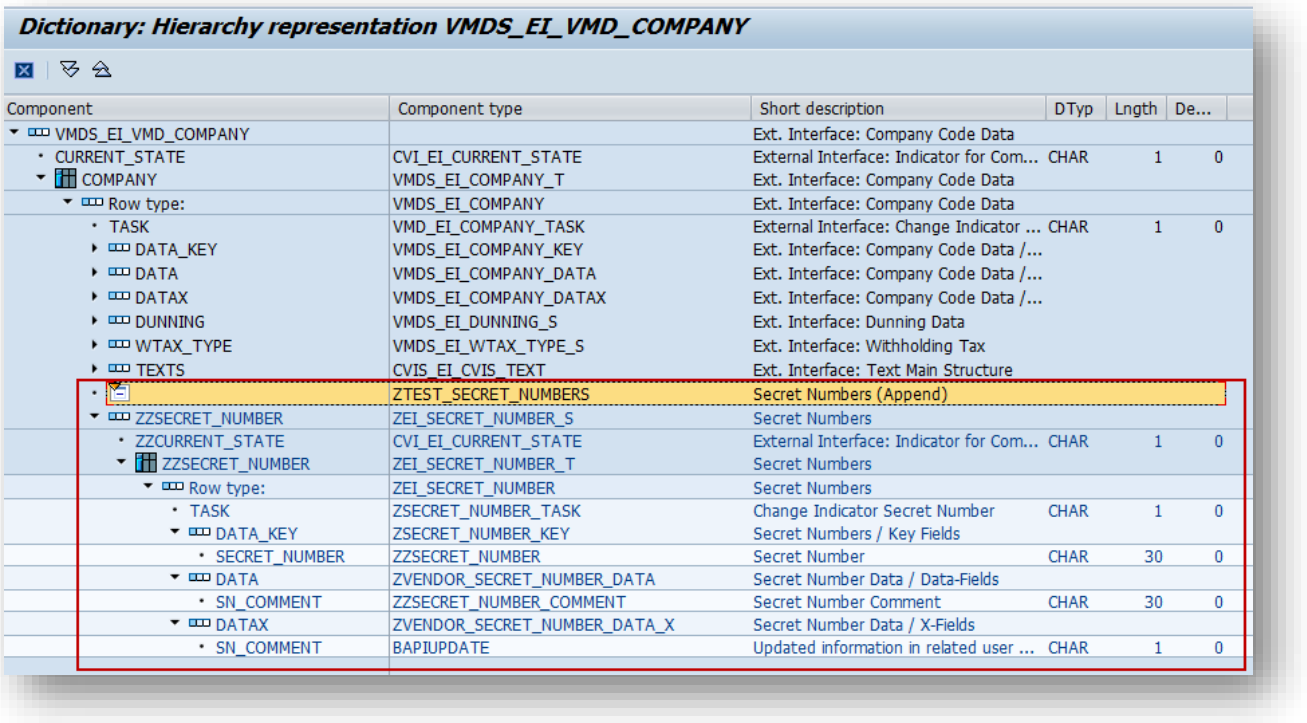
Name	Description
ZEI_SECRET_NUMBER	Secret Numbers
ZEI_SECRET_NUMBER_S	Secret Numbers

ZSECRET_NUMBER_KEY	Secret Numbers / Key Fields
ZVENDOR_SECRET_NUMBER_DATA	Secret Number Data / Data-Fields
ZVENDOR_SECRET_NUMBER_DATA_X	Secret Number Data / X-Fields
ZTEST_SECRET_NUMBERS	Secret Numbers (Append)

Data Elements	
Name	Description
ZSECRET_NUMBER_TASK	Change Indicator Secret Number
ZZSECRET_NUMBER	Secret Number
ZZSECRET_NUMBER_COMMENT	Secret Number Comment

Domains	
Name	Description
ZSECRET_NUMBER_TASK	Change Indicator Secret Numbers
ZZSECRET_NUMBER	Secret Number
ZZSECRET_NUMBER_COMMENT	Secret Number Comment

Now you are ready to extend structure **VMDS_EI_COMPANY** by creating an append called **ZTEST_SECRET_NUMBERS** using component type **ZEI_SECRET_NUMBER_S** as shown below.

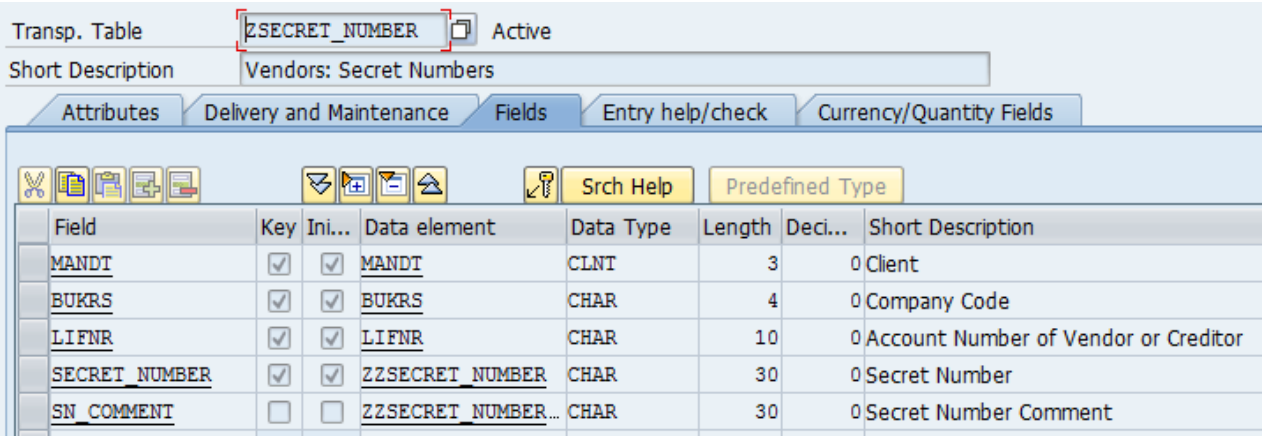


4.2 Create table for storing secret numbers in the reuse area

You still need a table where the secret numbers are stored after activation of the change request.

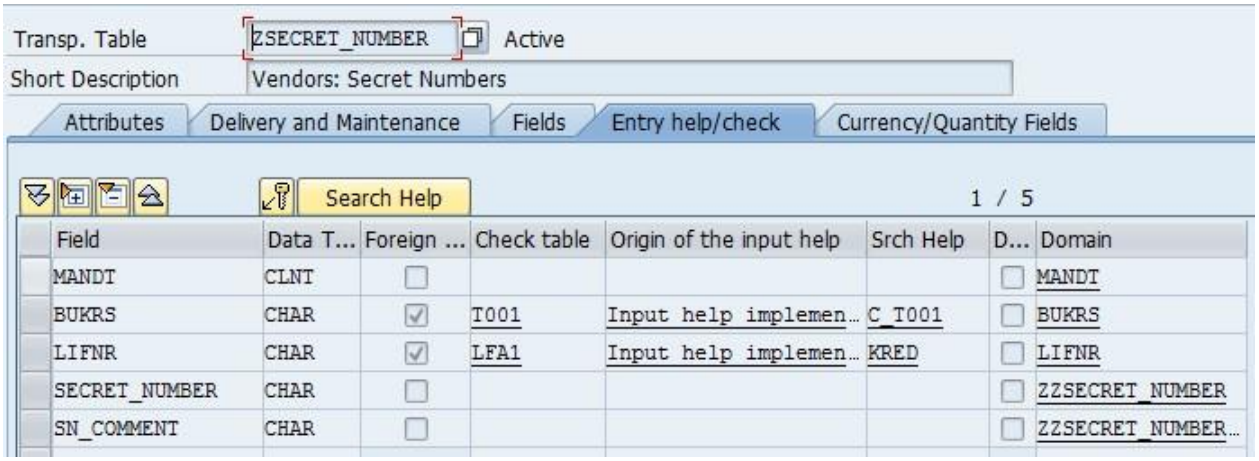
Database Tables	
Name	Description
ZSECRET_NUMBER	Table used to store the secret numbers entered via MDG change request.

ZSECRET_NUMBER (Fields)



Field	Key	Ini...	Data element	Data Type	Length	Deci...	Short Description
MANDT	☑	☑	MANDT	CLNT	3	0	Client
BUKRS	☑	☑	BUKRS	CHAR	4	0	Company Code
LIFNR	☑	☑	LIFNR	CHAR	10	0	Account Number of Vendor or Creditor
SECRET_NUMBER	☑	☑	ZZSECRET_NUMBER	CHAR	30	0	Secret Number
SN_COMMENT	☐	☐	ZZSECRET_NUMBER...	CHAR	30	0	Secret Number Comment

ZSECRET_NUMBER (Entry help / check)

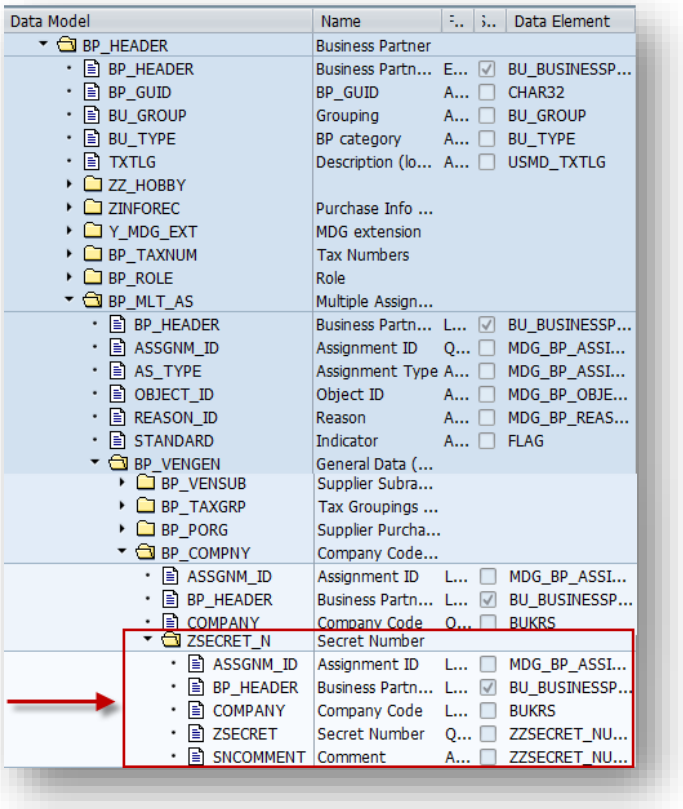


Field	Data T...	Foreign ...	Check table	Origin of the input help	Srch Help	D...	Domain
MANDT	CLNT	☐				☐	MANDT
BUKRS	CHAR	☑	T001	Input help implemen...	C_T001	☐	BUKRS
LIFNR	CHAR	☑	LFA1	Input help implemen...	KRED	☐	LIFNR
SECRET_NUMBER	CHAR	☐				☐	ZZSECRET_NUMBER
SN_COMMENT	CHAR	☐				☐	ZZSECRET_NUMBER...

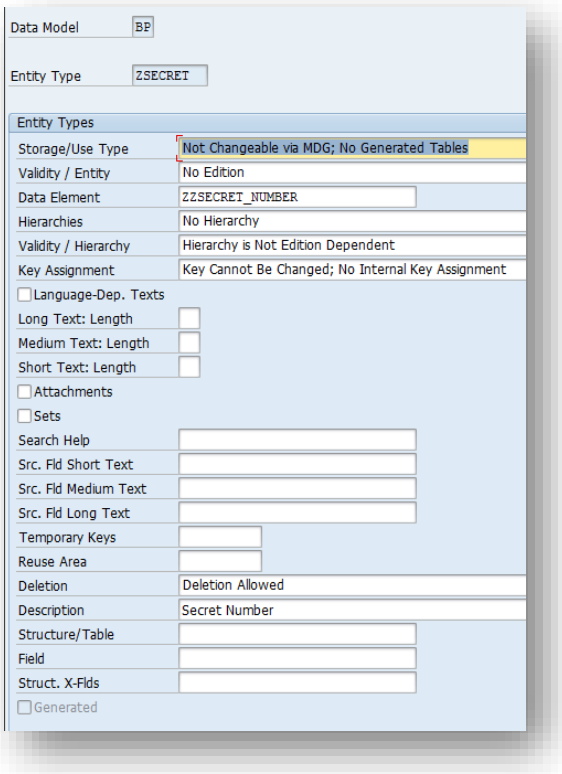
4.3 Extend the MDG-S data model with your new table like entity

While change requests are in process, your secret numbers are stored in the MDG staging area. Therefore,you must extend the BP data model accordingly. The necessary steps are outlined in this section.

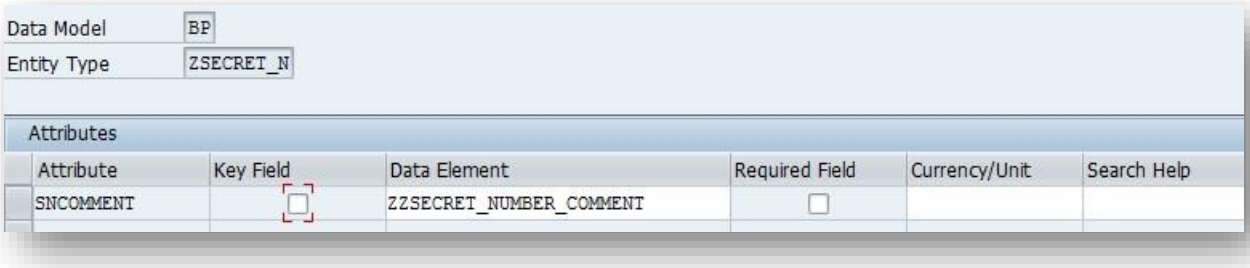
You start by creating two new entity types; one for your secret number and another for the associated description. In the next step, you establish a relationship between these entity types and to the BP_COMPANY entity type. In the last step of this section, you generate additional data model-specificstructures.



4.3.1 Create New Entity Types



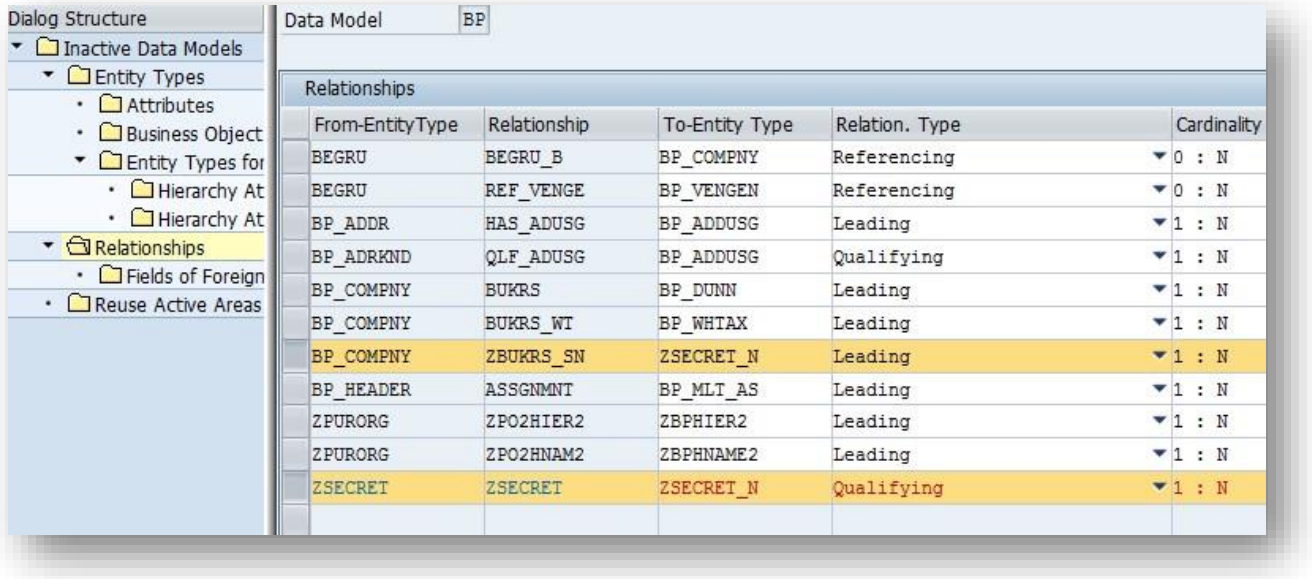
4.3.2 Relationships



Data Model: BP
Entity Type: ZSECRET_N

Attributes

Attribute	Key Field	Data Element	Required Field	Currency/Unit	Search Help
SNCOMMENT	☐	ZZSECRET_NUMBER_COMMENT	☐		



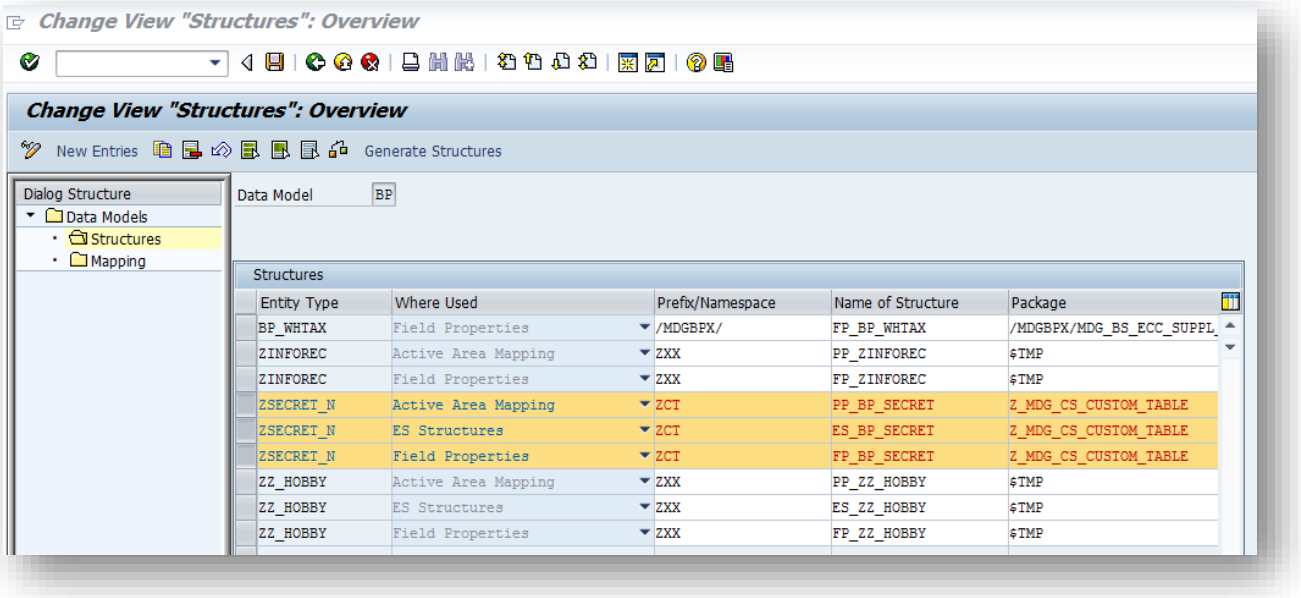
Dialog Structure
- Inactive Data Models
  - Entity Types
    - Attributes
    - Business Object
    - Entity Types for
      - Hierarchy At
      - Hierarchy At
  - Relationships
    - Fields of Foreign
  - Reuse Active Areas

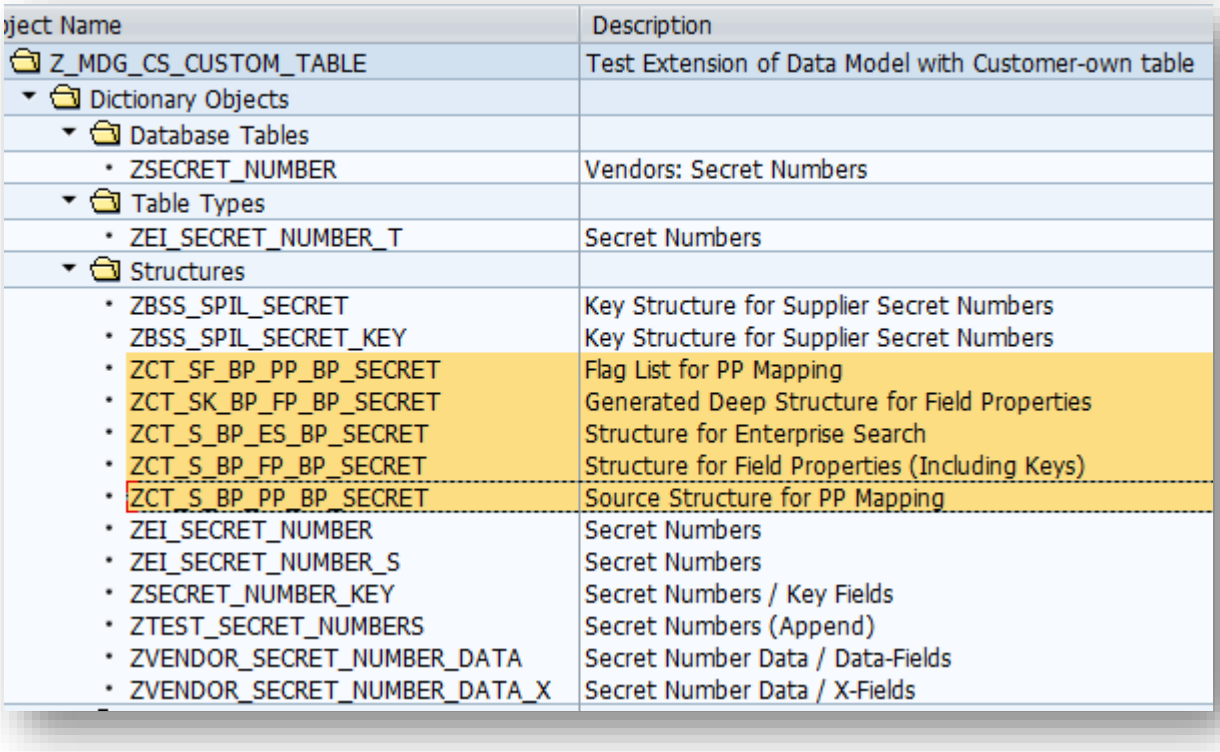
Data Model: BP

Relationships

From-EntityType	Relationship	To-Entity Type	Relation. Type	Cardinality
BEGRU	BEGRU_B	BP_COMPNY	Referencing	0 : N
BEGRU	REF_VENGE	BP_VENGEN	Referencing	0 : N
BP_ADDR	HAS_ADUSG	BP_ADDUSG	Leading	1 : N
BP_ADRKND	QLF_ADUSG	BP_ADDUSG	Qualifying	1 : N
BP_COMPNY	BUKRS	BP_DUNN	Leading	1 : N
BP_COMPNY	BUKRS_WT	BP_WHTAX	Leading	1 : N
BP_COMPNY	ZBUKRS_SN	ZSECRET_N	Leading	1 : N
BP_HEADER	ASSGNMNT	BP_MLT_AS	Leading	1 : N
ZPURORG	ZPO2HIER2	ZBPHIER2	Leading	1 : N
ZPURORG	ZPO2HNAM2	ZBPHNAME2	Leading	1 : N
ZSECRET	ZSECRET	ZSECRET_N	Qualifying	1 : N

4.3.3 Generate Data Model-Specific Structures





4.4 Create and Implement Your Own Handler Class

4.4.1 Create Your Own Handler Class

To create your own handler class by inheriting from class **CL_MDG_BS_ECC_HANDLER**, follow the steps of How To Guide Create and Register a Custom Handler Class.

4.4.2 Implement Your Own Handler Class

Since the superclass is abstract, create redefinitions for all methods belonging to the interface **IF_MDG_BS_BP_ACCESS_HANDLER**. The redefinitions itself can be empty at first. It is sufficient to implementonly those methods that are required to fulfill the needs of your process.

In this scenario you must implement the methods listed below. In addition, you must create a function modulethat we call **SAVE_ADDITIONAL_OBJECT_DATA** in our example with the option "**IN UPDATE TASK**" to ensure transactional consistency.

A detailed source code example is provided in the Appendix section Source Code for Custom Handler Class.

4.4.3 Re-implement Methods

You can look at the code from **CL_MDG_BS_SUPPL_HANDLER** to get an idea what should be done in these methods.

Name	Description
MAP_DATA_2API	Mapping data to the API
MAP_DATA_2STA	Mapping data to staging
PREPARE_EI_HEADER_MAP_2API	
READ_OBJECT_DATA	Read data
SAVE_ADDITIONAL_OBJECT_DATA	Save data
SORT_ENTITIES	Sort data

Function Modules

Name	Description
Z_SECRET_NUMBER_UPDATES	Save data to Z* table

4.5 Extend the genIL model

In preparation of the User Interface extensions, you need to create an enhancement of the genIL model BUPA_CUSP.

4.5.1 Create Data Dictionary Objects

During the genIL model extension you will need to provide a structure for attributes and one for keys. Thenames of these structures in our example are listed in the table below. For details of the individual DDIC objects refer to the Appendix Data Dictionary Objects for genIL Model Extension.

Structures	
Name	Description
ZBSS_SPIL_SECRET	Attribute Structure for Supplier SecretNumbers.
ZBSS_SPIL_SECRET_KEY	Key Structure for Supplier Secret Numbers.

4.5.2 genIL Component Class for Custom Table

1. Create a new class **ZCL_BS_GENIL_SUPPLIER_CUST_TAB** and use **CL_BS_GENIL_CUSTOMER** asthe superclass.
2. Reimplement the following methods:

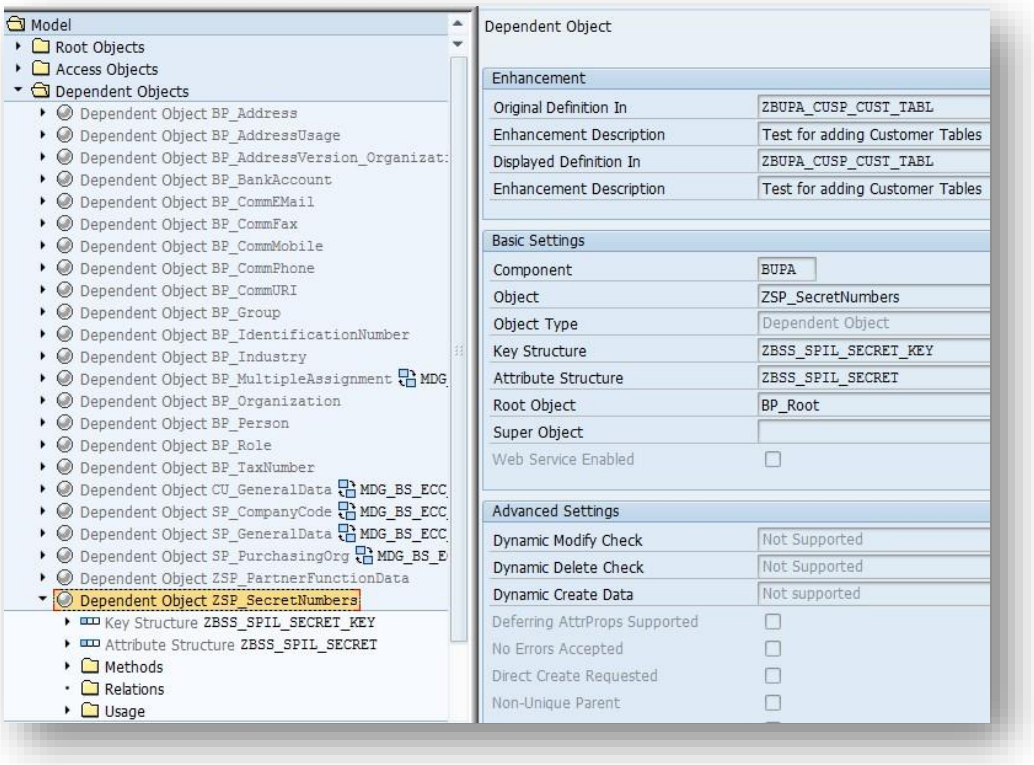
Name	Description
IS_CHILD_CREATE_ALLOWED	Determines if it is allowed to createdependent objects.
TRANSFORM_TO_ENTITY_KEY	Transforms an object key to an entitykey.

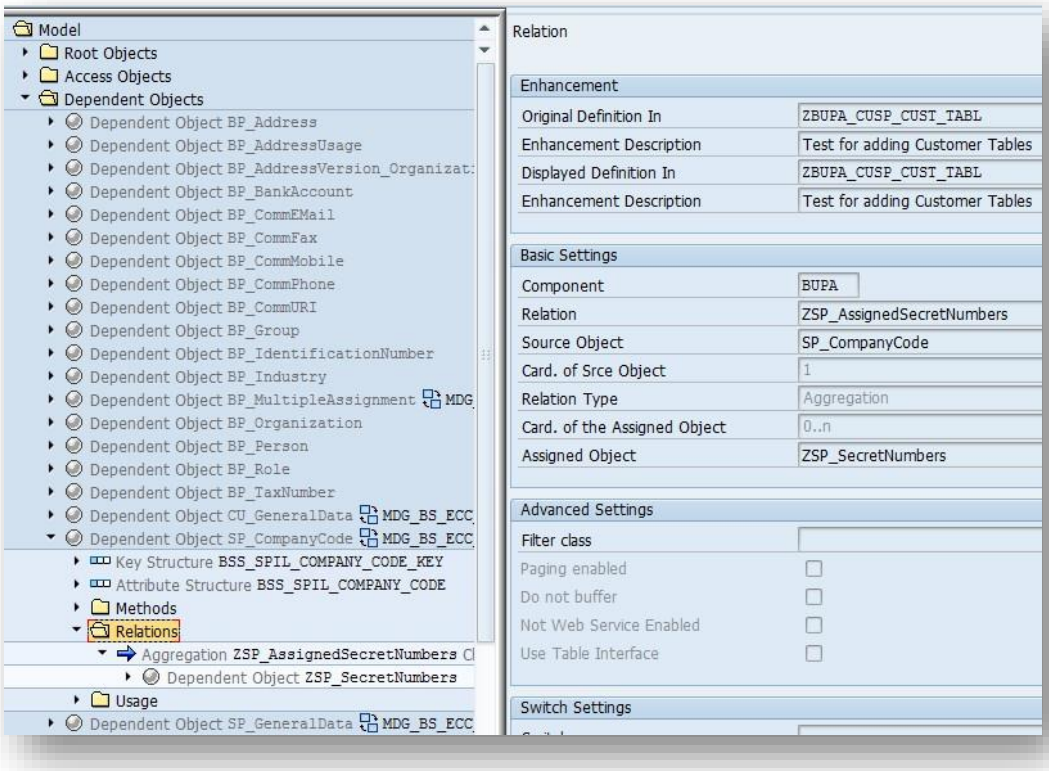
   A detailed source code example is provided in the Appendix section Source Code for genIL Model Class.

3. Activate your changes.

4.5.3 genIL Model Enhancement for Custom Table

1. Start SAP backend with transaction GENIL_MODEL_BROWSER.
2. Create a new enhancement. Let's call it ZBUPA_CUSP_CUST_TABL. In the Create Enhancement dialog enter BUPA_CUSP as Super enhancement.
3. In the Component Details screen enter implementation class ZCL_BS_GENIL_SUPPLIER_CUST_TAB (the class that you have created in the previous step).
4. Create a new Dependent Object with the name ZSP_SecretNumbers.
5. Create a new Relation below the SP_CompanyCode element. As a name for the relation we use ZSP_AssginedSecretNumbers.



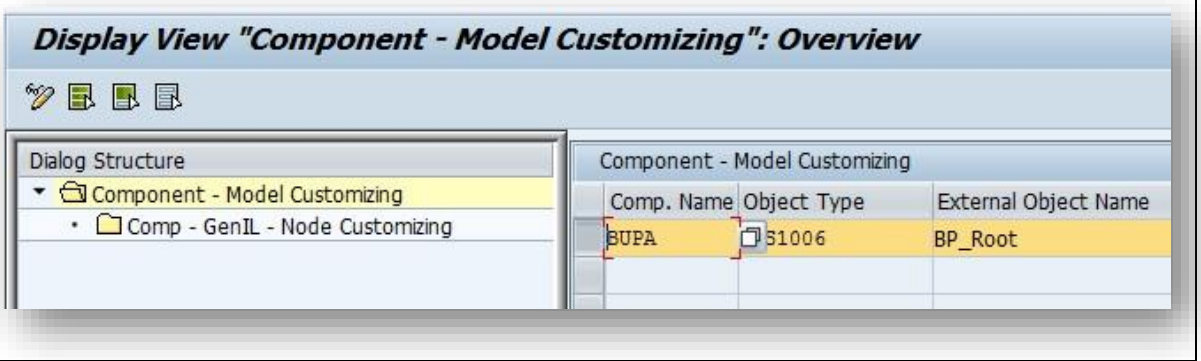


4.5.4 Connect the MDG Data Model with the genIL Data Model

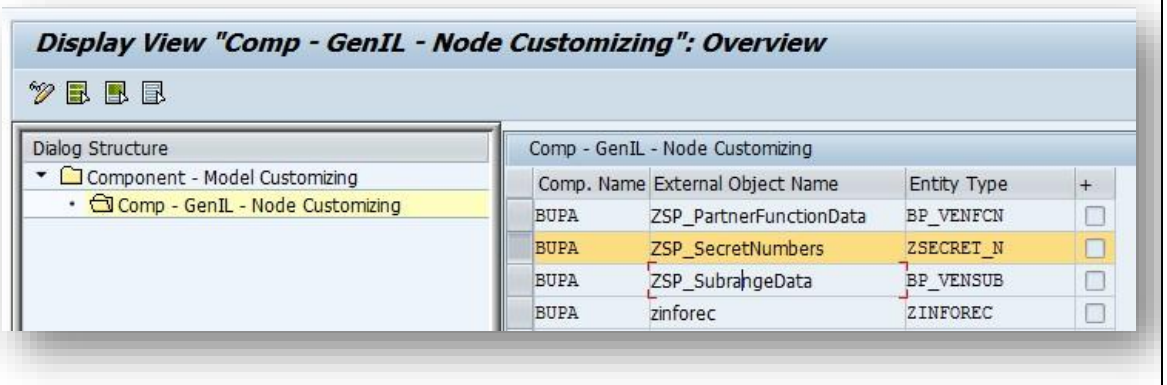
1. Start the view cluster maintenance with transaction **SM34**
2. Enter View Cluster **VC_MDG_BS_GENIL_C**
3. For the ***Model Customizing*** create a new entry with the following values (unless there is one such entry in the list already):
   - Component Name: BUPA
   - Object Type: BUS1006
   - External Object Name: BP_Root.
4. For the ***Node Customizing*** create a new entry with the following values:
   - Component Name: BUPA
   - *External Object Name*: ZSP_SecretNumbers
   - *Entity Type*: ZSECRET_N.

   Note that if you have chosen other names in your GenIL or MDG models you need to use those names.

Component – Model Customizing



Component – Node Customizing



4.6 Extending the MDG-S User Interface

The remaining steps focus on extending the MDG-S user interface.

4.6.1 GUIBB Feeder for Supplier Secret Numbers

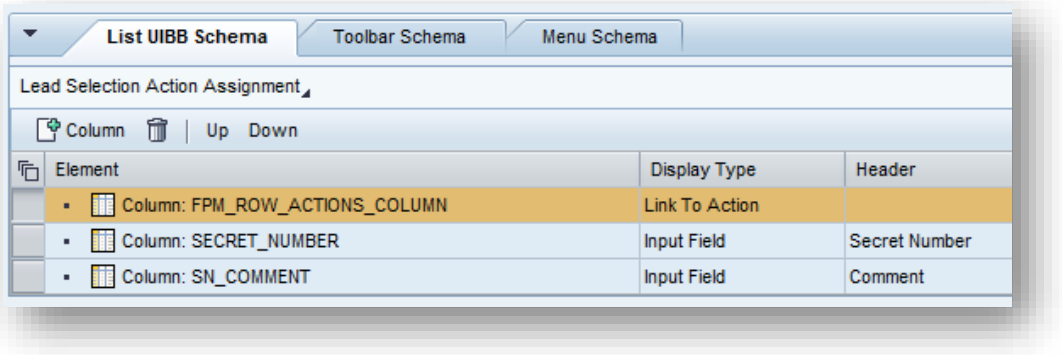
6. Create a new class called **ZCL_BS_SP_GUIBB_SECRET_NUMBERS**using **CL_BS_BP_GUIBB_LIST**assuperclass. You do not need to make any further changes to the class at this stage. For more information, see Extensibility Options for SAP Master Data Governance -> Customer / Supplier Data -
   ≥ Create or Redefine a UI Feeder Class.
7. Activate your changes.

4.6.2 Copy Component Configuration FPM_LIST_UIBB

1. Start transaction **SE80**.
2. Search for Web Dynpro Component **FPM_LIST_UIBB**
3. Locate and open the Component Configuration **FPM_LIST_UIBB_TEMPLATE**
4. Use the button Start Configurator to launch the configurator for **FPM_LIST_UIBB_TEMPLATE**
5. In the configurator use the button Copy Configuration. Call the new configuration ZBS_SP_SECRET_NUMBERS.
6. Start the component configurator for **ZBS_SP_SECRET_NUMBERS**.
7. In the **feeder**class parameters dialogue enter the following values

Feeder Class	**ZCL_BS_SP_GUIBB_SECRET_NUMBERS**
Component Name	**BUPA**
Object Name	**ZSP_SecretNumbers**
Editable	yes

8. Add the following columns to your List UIBB



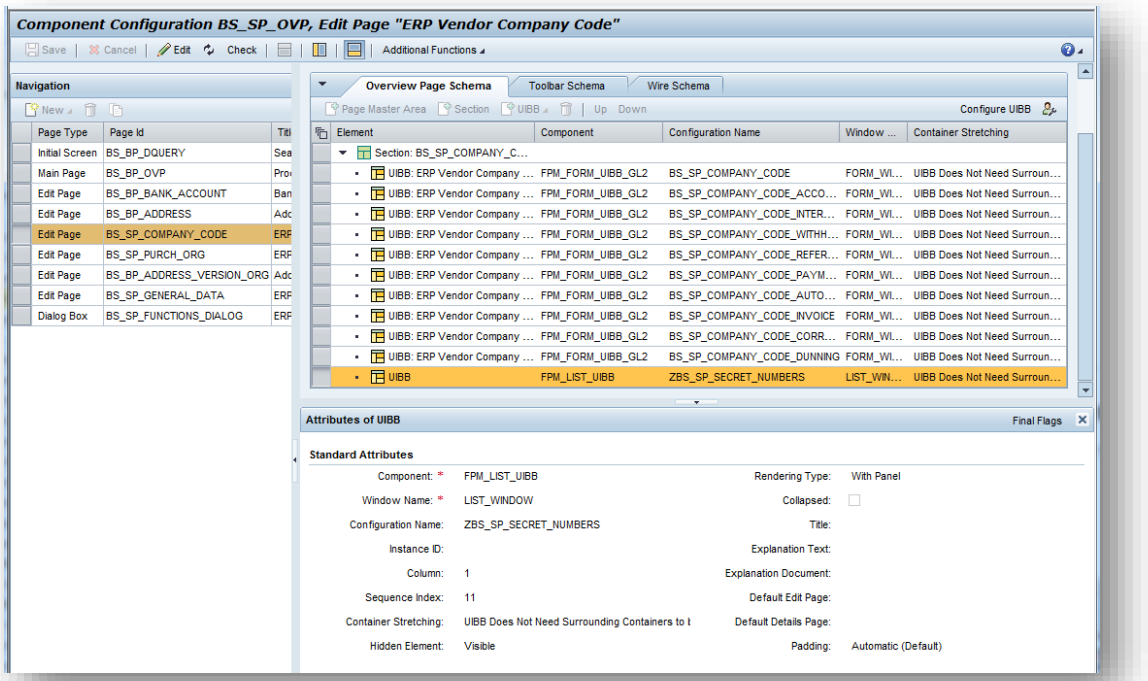
9. For the action column enter the following action assignments in the attributes **FPM_BOL_ROW_DELETE / ~Icon/Delete**

4.6.3 Enhancement of UI Configuration

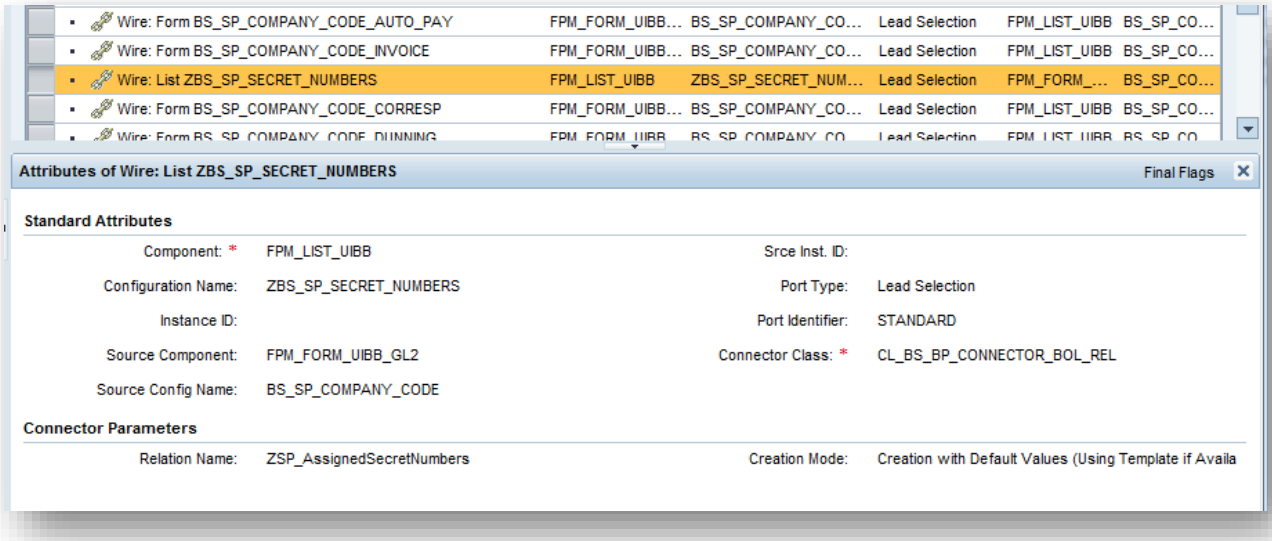
In this final task you create an enhancement of the Supplier Overview Page Floorplan (**BS_SP_OVP**) toinclude the List UIBB you have created in the previous step.

1. Start transaction **SE80**
2. Open package **MDG_BS_ECC_SUPPLIER_BOLUI**Below the Web Dynpro tree node open component configuration **BS_SP_OVP**and start theconfigurator. Press the button “Continue in Change Mode”

3. From the top level menu select ***Additional Functions -> Enhance*** to create an Enhancement of the existing UI configuration. Provide a name for the enhancement for example **ZSECRET_NUMBERS_BS_OVP_SP**
4. Switch on the display of the navigation panel on the left hand side. In the navigation panel select BS_SP_COMPAMY_CODE
5. Add your List **UIBB ZBS_SP_SECRET_NUMBERS** to the Page Schema



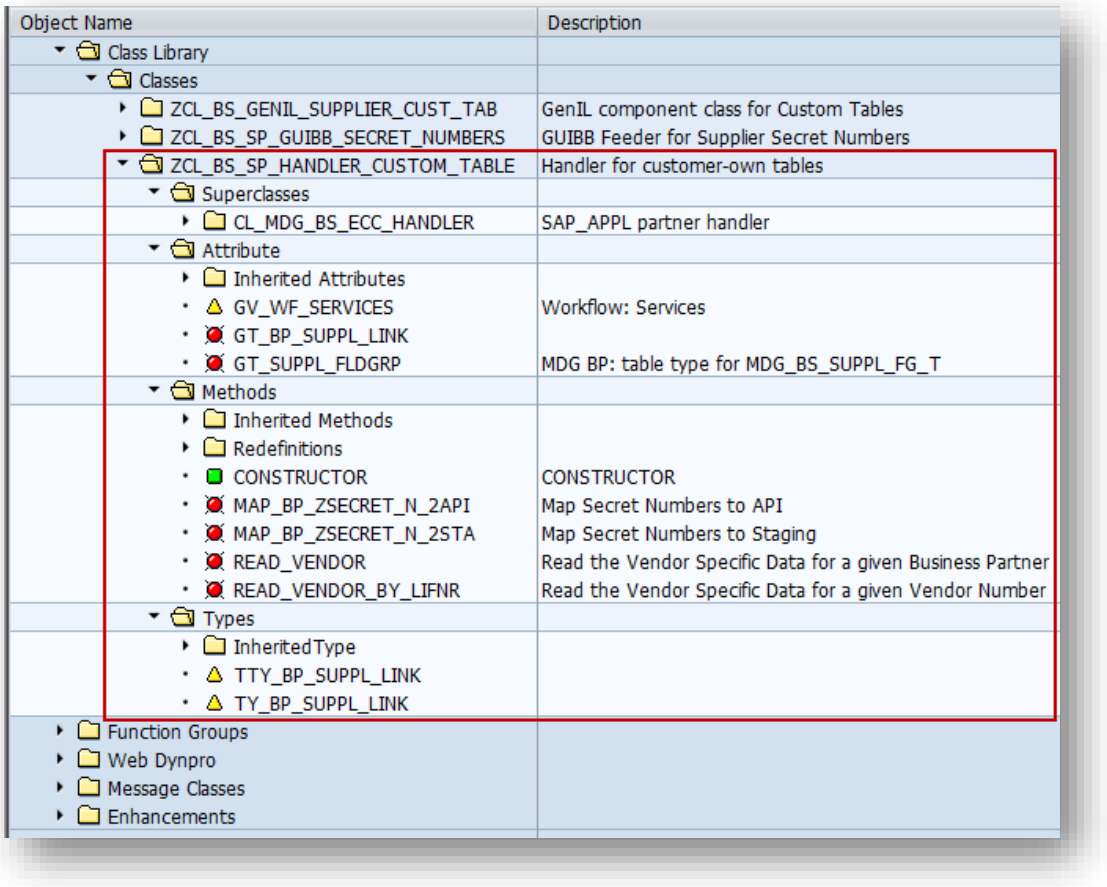
6. Add a reference to the List UIBB to the Wiring



7. Save your changes
8. Result: You have extended MDG-S to handle an additional table like entity.

5 Appendix

5.1 Source Code for Custom Handler Class



5.1.1 Re-Implement methods

MAP_DATA_2API

```
method IF_MDG_BS_BP_ACCESS_HANDLER~MAP_DATA_2API.



  DATA: ls_cvi_ei_extern TYPE cvis_ei_extern.
*       ls_suppl_ext     TYPE mdg_bs_bp_s_mlt_as_suppl     .

  FIELD-SYMBOLS:
          <lt_mlt_as>      TYPE mdg_bs_bp_tt_mlt_as,
          <ls_mlt_as>      TYPE mdg_bs_bp_s_mlt_as,
          <lt_data>        TYPE mdg_bs_bp_tt_mlt_as_suppl,
```

```
            <ls_data>         TYPE vmds_ei_extern,
            <ls_ext_suppl>    TYPE VMDS_EI_EXTERN,
            <ls_bp_header>    TYPE bus_ei_extern,
            <ls_as_id>        TYPE mdg_bp_assignment_id,
            <ls_ext_as_id>    TYPE mdg_bp_assignment_id,
            <ls_suppl>          TYPE mdg_bs_bp_s_mlt_as_suppl. " vmds_ei_extern.

  ASSIGN COMPONENT 'ASSGNM_ID'  OF STRUCTURE is_data     TO <ls_as_id>.
  ASSIGN COMPONENT 'VENDOR'     OF STRUCTURE cs_data_ext TO <ls_ext_suppl> .

  CHECK  <ls_ext_suppl> IS ASSIGNED.

*      ls_cvi_ei_extern-partner-header-
object_instance_bpartner = not available here...
  ls_cvi_ei_extern-vendor = <ls_ext_suppl> .

  CASE iv_entity.


    WHEN 'ZSECRET_N'.
      CALL METHOD me->MAP_BP_ZSECRET_N_2API
        EXPORTING
          is_data    = is_data
          iv_task    = iv_task
          iv_fname   = iv_fname
        CHANGING
          cs_suppl   = ls_cvi_ei_extern
          ct_return = ct_return
          cv_xchange = cv_xchange.

  WHEN OTHERS.

ENDCASE.


  IF  <ls_ext_suppl> IS ASSIGNED.
    <ls_ext_suppl>         = ls_cvi_ei_extern-vendor.
    ASSIGN COMPONENT 'ASSIGNMENT_ID' OF STRUCTURE cs_data_ext TO <ls_ext_as_id>.
    IF sy-subrc = 0.
      <ls_ext_as_id> = <ls_as_id>.
```

```
    ENDIF.
  ENDIF.

endmethod.
```

MAP_DATA_2STA

```
method IF_MDG_BS_BP_ACCESS_HANDLER~MAP_DATA_2STA.

  DATA: ls_cvi_ei_extern TYPE cvis_ei_extern,
        lv_as_id TYPE mdg_bp_assignment_id.


FIELD-SYMBOLS:
           <lt_mlt_as>        TYPE ANY TABLE,
           <ls_mlt_as>        TYPE any,
           <lt_data>          TYPE mdg_bs_bp_tt_mlt_as_suppl,
           <ls_data>          TYPE mdg_bs_bp_s_mlt_as_suppl,
           <ls_bp_header>     TYPE bus_ei_extern,
           <ls_data_ret>      TYPE any,
           <lv_as_id_ret>     TYPE any,
           <lv_as_id>         TYPE any.


  *     special handling for multiple assignments:
  *     in case a multiple assignment record   for standard vendor was simulated (ifcvi_vend_link exist but no
entry
  *     in table MDG_MLT_ASSGNMNT) then only this record is bufferd by customer handler and has to be
mapped here, that's why entity 'BP_MLT_AS'
  *     has to be processed here additionally
  IF iv_entity = 'BP_MLT_AS'.
    ASSIGN COMPONENT 'MLT_ASSIGNMENTS' OF STRUCTURE is_data TO <lt_mlt_as>.
    CALL METHOD me->map_bp_mlt_as_2sta
      EXPORTING
        it_mlt_as = <lt_mlt_as>
      CHANGING
```

```
          ct_data    = ct_data
          ct_return = ct_return.

  RETURN.
ENDIF.

  *    handling of all other entities
    ASSIGN COMPONENT 'PARTNER' OF STRUCTURE is_data TO <ls_bp_header>.
    ASSIGN COMPONENT 'VENDORS' OF STRUCTURE is_data TO <lt_data>.

    IF <lt_data> IS ASSIGNED.
      LOOP AT <lt_data> ASSIGNING <ls_data>.

  *         build CVI structure for mapping ls_cvi_ei_extern-
        partner = <ls_bp_header>. ls_cvi_ei_extern-vendor =
        <ls_data>-vendor.
        ASSIGN COMPONENT 'ASSIGNMENT_ID' OF STRUCTURE <ls_data> TO <lv_as_id>.
        IF sy-subrc = 0.
          lv_as_id = <lv_as_id>.

        ENDIF.

        CASE iv_entity.

          WHEN 'ZSECRET_N'.
            CALL METHOD me->MAP_BP_ZSECRET_N_2STA
              EXPORTING
                is_suppl         = ls_cvi_ei_extern
                iv_assignment_id = lv_as_id

              CHANGING
                ct_data          = ct_data
                ct_return        = ct_return.


        ENDCASE

      .ENDLOOP.
    ENDIF.

endmethod.
```

PREPARE_EI_HEADER_MAP_2API

```
method IF_MDG_BS_BP_ACCESS_HANDLER~PREPARE_EI_HEADER_MAP_2API.

FIELD-SYMBOLS:
        <ls_suppl_ext>  TYPE cvis_ei_extern,
        <bp_suppl_link> TYPE ty_bp_suppl_link.



DATA:
      ls_suppl_ext    TYPE cvis_ei_extern.



FIELD-SYMBOLS:
       <lv_partner>         TYPE bu_partner,
       <ls_ecc_ext>         TYPE mdg_bs_bp_s_ecc_extern,
       <ls_data_ext_db>     TYPE mdg_bs_bp_s_ecc_extern,
       <ls_vendor_db>       TYPE mdg_bs_bp_s_mlt_as_suppl,
       <lv_kunnr>           TYPE kunnr,
       <ls_ecc_ext_suppl>   TYPE mdg_bs_bp_s_mlt_as_suppl,
       <lv_assignment_id>   TYPE mdg_bp_assignment_id,
       <ls_ecc_mlt_as>      TYPE mdg_bs_bp_s_mlt_as_mem,
       <ls_ext>             TYPE mdg_bs_bp_s_mlt_as_suppl.



DATA:
      ls_cust_ext         TYPE cvis_ei_extern,
      ls_ecc_ext          TYPE mdg_bs_bp_s_ecc_extern,
      lr_data             TYPE REF TO data,
      ls_ecc_ext_suppl    TYPE mdg_bs_bp_s_mlt_as_suppl,
      ls_ecc_mlt_as       TYPE mdg_bs_bp_s_mlt_as_mem.
```

```
  CHECK iv_entity = 'ZSECRET_N'.


* Supplier Switch for EHP6 has to be active
  IF cl_mdg_bs_suppl_switch_check=>mdg_bs_ecc_supplier_switch_02( ) EQ abap_true
.

    ASSIGN COMPONENT 'BP_HEADER'  OF STRUCTURE is_data TO <lv_partner>.
    ASSIGN COMPONENT 'ASSGNM_ID'  OF STRUCTURE is_data TO <lv_assignment_id>.

*   check whether record with bp number already exists, if not create one
    READ TABLE ct_data_ext
      WITH KEY ('PARTNER-HEADER-OBJECT_INSTANCE-BPARTNER') = <lv_partner>
     ASSIGNING <ls_ecc_ext>.

*   should always be provided by BP_Handler
    IF sy-subrc <> 0.
*     create BP record
      CLEAR ls_ecc_ext.
      ls_ecc_ext-partner-header-object_instance-bpartner = <lv_partner>.

      READ TABLE gt_ecc_extern_db
      ASSIGNING <ls_data_ext_db>
       WITH KEY partner-header-object_instance-bpartner = <lv_partner>.

      IF sy-subrc = 0.
        ls_ecc_ext-partner-header-object_task = gc_upd.
      ELSE.
        ls_ecc_ext-partner-header-object_task = gc_ins.
      ENDIF.
      INSERT ls_ecc_ext INTO TABLE ct_data_ext ASSIGNING <ls_ecc_ext>.
    ENDIF.


*   check if vendor record was already mapped
```

```
    READ TABLE <ls_ecc_ext>-vendors
         WITH KEY assignment_id = <lv_assignment_id>  ASSIGNING <ls_ecc_ext_supp
l>.


*    create new supplier entry
    IF sy-subrc <> 0.

      ls_ecc_ext_suppl-assignment_id  = <lv_assignment_id>.

*      determine LIFNR (may already exist on DB)
      READ TABLE gt_ecc_extern_db
           ASSIGNING <ls_data_ext_db>
            WITH KEY partner-header-object_instance-bpartner = <lv_partner>.
*      Partner already exist on DB
      IF sy-subrc = 0.
        READ TABLE <ls_data_ext_db>-vendors ASSIGNING <ls_vendor_db>
             WITH KEY assignment_id = <lv_assignment_id>.
*        supplier already exist on DB
        IF sy-subrc = 0.
*          fill suplier ID
          ls_ecc_ext_suppl-vendor-header-object_instance-lifnr = <ls_vendor_db>-
vendor-header-object_instance-lifnr.
          ls_ecc_ext_suppl-vendor-header-object_task  = gc_upd.
        ELSE.
*          supplier doesn't exist on DB, must be Insert
          ls_ecc_ext_suppl-vendor-header-object_task  = gc_ins.
        ENDIF.
      ENDIF.

      INSERT ls_ecc_ext_suppl INTO TABLE <ls_ecc_ext>-
vendors ASSIGNING  <ls_ecc_ext_suppl>.
    ENDIF. "create new supplier target record

    GET REFERENCE OF <ls_ecc_ext_suppl> INTO er_data_ext.

  ELSE. "switch check
*    EHP5 is active
    ASSIGN COMPONENT 'BP_HEADER' OF STRUCTURE is_data TO <lv_partner>.
```

```
*       Check whether record with bp number already exists, if not create one
    READ TABLE ct_data_ext
      WITH KEY ('PARTNER-HEADER-OBJECT_INSTANCE-BPARTNER') = <lv_partner>
     ASSIGNING <ls_suppl_ext>.

    IF sy-subrc <> 0.
*       Create record
      ls_suppl_ext-partner-header-object_instance-bpartner = <lv_partner>.

*       Get supplier number from Business Partner ID
      READ TABLE gt_bp_suppl_link
        WITH KEY bpartner = <lv_partner>
       ASSIGNING <bp_suppl_link>.

      IF sy-subrc EQ 0.
*         Check whether record with bp number already exists, if not create one
        READ TABLE ct_data_ext
          WITH KEY ('VENDOR-HEADER-OBJECT_INSTANCE-LIFNR') = <bp_suppl_link>-
supplier
         ASSIGNING <ls_suppl_ext>.

        IF sy-subrc <> 0.
          ls_suppl_ext-vendor-header-object_instance-lifnr = <bp_suppl_link>-
supplier.
*           Default value for Update task is UPDATE
*           If it is an INSERT or DELETE than an Header record will overwrite the task
          ls_suppl_ext-vendor-header-object_task = gc_upd.
          ls_suppl_ext-partner-header-object_task = gc_upd.

        ENDIF.
      ENDIF.

      INSERT ls_suppl_ext INTO TABLE ct_data_ext ASSIGNING <ls_suppl_ext>.

    ELSE. "record found
      IF <ls_suppl_ext>-vendor-header-object_instance-lifnr IS INITIAL.

        READ TABLE gt_bp_suppl_link
```

```
          WITH KEY bpartner = <lv_partner>
         ASSIGNING <bp_suppl_link>.

        IF sy-subrc EQ 0.
          <ls_suppl_ext>-vendor-header-object_instance-lifnr = <bp_suppl_link>-
supplier.
          <ls_suppl_ext>-vendor-header-object_task = gc_upd.
        ENDIF.


      ENDIF.
    ENDIF.

    GET REFERENCE OF <ls_suppl_ext> INTO er_data_ext.
  ENDIF. "switch check


endmethod.
```

READ_OBJECT_DATA

```
method IF_MDG_BS_BP_ACCESS_HANDLER~READ_OBJECT_DATA.

  DATA: lt_supplier            TYPE cvis_ei_extern_t,
        ls_as_vendor           TYPE cvis_ei_extern,
        lt_mlt_as              TYPE mdg_bs_bp_tt_mlt_as,
        lt_mlt_as_old          TYPE mdg_bs_bp_tt_mlt_as_mem,
        lt_mlt_as_ret          TYPE mdg_bs_bp_tt_mlt_as_mem,
        ls_mlt_as              TYPE mdg_bs_bp_s_mlt_as,
        ls_mlt_as_old          TYPE mdg_bs_bp_s_mlt_as_mem,
        ls_mlt_as_db           TYPE mdg_bs_bp_s_mlt_as_target,
        lt_return              TYPE bapiret2_t,
        lv_lifnr               TYPE lifnr.


  FIELD-SYMBOLS:
        <ls_cvis_data_db>         TYPE cvis_ei_extern,
        <ls_supplier>             TYPE cvis_ei_extern,
        <ls_as_supplier>          TYPE cvis_ei_extern,
        <ls_mlt_as_db>            TYPE mdg_bs_bp_s_mlt_as_target.
```

```
DATA: lt_ecc_extern_db TYPE mdg_bs_bp_tt_ecc_extern,
      ls_ecc_extern_db TYPE mdg_bs_bp_s_ecc_extern,
      ls_vendor        TYPE mdg_bs_bp_s_mlt_as_suppl,
      lv_kunnr         TYPE kunnr.


FIELD-SYMBOLS:
      <ls_ecc_extern_db>   TYPE mdg_bs_bp_s_ecc_extern,
      <ls_mlt_as_old>      TYPE mdg_bs_bp_s_mlt_as_mem,
      <ls_idlist>          TYPE bus_ei_instance.

data: lt_ZSECRET_NUMBER type table of ZSECRET_NUMBER,
      ls_ZSECRET_NUMBER like line of lt_ZSECRET_NUMBER.

FIELD-SYMBOLS: <ls_extern_db>     like line of GT_ECC_EXTERN_DB,
               <ls_vendor>        type MDG_BS_BP_S_MLT_AS_SUPPL,
               <ls_company>       type line of VMDS_EI_COMPANY_T,
               <ls_secret_number> type ZEI_SECRET_NUMBER.

* global DB tables are filled here
* EHP5: gt_cvis_data_db -> contains data of standard supplier
* EHP6: gt_ecc_extern_db -
> contains all assigned supplier (incl. multiple assignment data)
* (every handler has its own instance of these tables)

* NOTE: Multiple Assignment Handler reads all assignments, but in case a standard supplier is assigned to
BP
* and no record exists in table MDG_MLT_ASSGNMNT, this record has to be simulated here and will be
appended to DB state

* flag controls if data for all entities have to be read
CHECK iv_read_all IS NOT INITIAL.

*    assigned vendor to BP is read (can be only one per BP)
CALL METHOD me->read_vendor
  EXPORTING
    it_idlist    = it_idlist
  RECEIVING
    et_supplier = lt_supplier. "note: lt_supplier may have an entry even if no
```

```
  supplier exists (in case it_idlist has mor than"one entry), vendor
    strcuture is empty then

    LOOP AT it_idlist ASSIGNING  <ls_idlist> .
*         read all supplier assignments for BP from DB (standard and additional assignments)
*         buffer table of each handler has to store ALL assignments (supplier & customer)
*         in method 'read_object_data_by_entitythe entity BP_MLT_AS is handled byall handlers
      CALL METHOD go_mlt_as_api->get_list_mlt_assgnmnt
        EXPORTING
          iv_partner        = <ls_idlist>-bpartner
          iv_assignment_cat = gc_as_cat_suppl

        IMPORTING
          et_mlt_as         = lt_mlt_as " supplier assignments only
          et_message        = lt_return.

      lt_mlt_as_old = lt_mlt_as.
*         get data of standard customer
      READ TABLE lt_supplier WITH KEY partner-header-object_instance-
bpartner = <ls_idlist>-bpartner
                 ASSIGNING <ls_supplier>.


      IF <ls_supplier> IS NOT ASSIGNED.
        CONTINUE.

      ELSE.
        IF <ls_supplier>-vendor-header IS INITIAL.
          CONTINUE.
        ENDIF.
      ENDIF.

*         get corresponding assignment_id for standard supplier
      READ TABLE lt_mlt_as_old ASSIGNING <ls_mlt_as_old>
          WITH KEY partner    = <ls_idlist>-bpartner
                   standard   = abap_true.

      IF <ls_mlt_as_old> IS ASSIGNED. "should always be assigned
```

```
      ls_vendor-assignment_id = <ls_mlt_as_old>-assignment_id.
      ls_vendor-vendor = <ls_supplier>-vendor.
*         vendor does not have standard assigment/we simulate one/will be writtenwhen activating/
assignment ID of
*         standard supplieris always '1'
    ELSE.
      ls_vendor-assignment_id       = gc_standard_as_id_suppl. " '1'
      ls_vendor-vendor              = <ls_supplier>-vendor.
*         create multiple assignment record
      ls_mlt_as_old-partner         = <ls_idlist>-bpartner.
      ls_mlt_as_old-assignment_id   = gc_standard_as_id_suppl.
      ls_mlt_as_old-assignment_cat  = gc_as_cat_suppl.
      ls_mlt_as_old-object_id       = ls_vendor-vendor-header-object_instance-
lifnr.
      ls_mlt_as_old-standard        = abap_true.
      ls_mlt_as_old-
updateflag            = gc_simulated_db_record.          "Simulated DB record/has to be written in save
later on

      APPEND ls_mlt_as_old TO lt_mlt_as_ret . " no append necessary for lt_mlt_a
s_old
    ENDIF.


*         check if there is already a record for BP in DB buffer table UNASSIGN
    <ls_ecc_extern_db>.                                          "1618668
    READ TABLE gt_ecc_extern_db ASSIGNING <ls_ecc_extern_db>
       WITH KEY partner-header-object_instance-bpartner = <ls_supplier>-partner-
header-object_instance-bpartner.

    IF <ls_ecc_extern_db> IS ASSIGNED.
*         check if vendor is already in DB buffer table
      READ TABLE <ls_ecc_extern_db>-vendors
          WITH KEY assignment_id                        = ls_vendor-assignment_id
                   vendor-header-object_instance-lifnr = <ls_supplier>-vendor-
header-object_instance-lifnr TRANSPORTING NO FIELDS.
      IF sy-subrc <> 0.
        APPEND ls_vendor TO <ls_ecc_extern_db>-vendors.
      ENDIF.
```

```
    ELSE.
*           create complete record
      ls_ecc_extern_db-partner-header-object_instance = <ls_idlist>.
      APPEND ls_vendor TO ls_ecc_extern_db-vendors.
      APPEND ls_ecc_extern_db TO gt_ecc_extern_db ASSIGNING <ls_ecc_extern_db>.
    ENDIF.

    IF <ls_ecc_extern_db> IS NOT ASSIGNED.
*       error
    ENDIF.



*       special handling for multiple assignments:
*       in case a multiple assignment record   for standard vendor was simulated (if cvi_vend_link exist but no entry
*       in table MDG_MLT_ASSGNMNT) then only this record is bufferd by customer handler and has to be stored here
    <ls_ecc_extern_db>-mlt_assignments = lt_mlt_as_ret.

*         read all additional suppliers
    LOOP AT lt_mlt_as_old INTO ls_mlt_as_old WHERE assignment_cat = gc_as_cat_suppl.

*     data of standard supplier has been already appended to table gt_ecc_extern_db, here only the additional assignments are treated
      IF ls_mlt_as_old-standard = abap_true.
        CONTINUE.
      ENDIF.

*           read data of assigned supplier
      CLEAR ls_as_vendor .
      lv_lifnr = ls_mlt_as_old-object_id.
      CALL METHOD me->read_vendor_by_lifnr
        EXPORTING
          iv_lifnr    = lv_lifnr
        IMPORTING
          es_supplier = ls_as_vendor.
```

```
      IF ls_as_vendor IS NOT INITIAL.

        READ TABLE <ls_ecc_extern_db>-vendors
                   WITH KEY assignment_id                          = ls_mlt_as_old-
assignment_id
                            vendor-header-object_instance-lifnr = ls_as_vendor-
vendor-header-object_instance-lifnr TRANSPORTING NO FIELDS.
        IF sy-subrc <> 0.
          ls_vendor-vendor        = ls_as_vendor-vendor.
          ls_vendor-assignment_id = ls_mlt_as_old-assignment_id.
          APPEND ls_vendor TO <ls_ecc_extern_db>-vendors.
        ENDIF.

      ENDIF.


    ENDLOOP. " LOOP AT lt_mlt_as_old
*       ENDLOOP. " LOOP AT lt_supplier
  ENDLOOP. "  LOOP AT it_id_list

*********************************************************************
* Part for Customer-own tables
  loop at GT_ECC_EXTERN_DB assigning <ls_extern_db>.
    loop at <ls_extern_db>-vendors assigning <ls_vendor>.
      loop at <ls_vendor>-VENDOR-COMPANY_DATA-COMPANY assigning <ls_company>.

        clear: lt_ZSECRET_NUMBER,
               ls_ZSECRET_NUMBER.
        SELECT * FROM ZSECRET_NUMBER appending table lt_ZSECRET_NUMBER
               WHERE  BUKRS  = <ls_company>-DATA_KEY-BUKRS

               AND    LIFNR  = <LS_VENDOR>-VENDOR-HEADER-OBJECT_INSTANCE-LIFNR.

        if not lt_ZSECRET_NUMBER is initial.
          <ls_company>-ZZSECRET_NUMBER-ZZCURRENT_STATE = 'X'.
        endif.

        loop at lt_ZSECRET_NUMBER into ls_ZSECRET_NUMBER.
          APPEND INITIAL LINE TO <ls_company>-ZZSECRET_NUMBER-
ZZSECRET_NUMBER ASSIGNING <ls_secret_number>.
```

```
          <ls_secret_number>-DATA_KEY-SECRET_NUMBER = ls_ZSECRET_NUMBER-
SECRET_NUMBER.
          <ls_secret_number>-DATA-SN_COMMENT        = ls_ZSECRET_NUMBER-
SN_COMMENT.
        endloop.

      endloop.
    endloop.
  endloop.
*****************************************************************

endmethod.
```

SAVE_ADDITIONAL_OBJECT_DATA

```
METHOD if_mdg_bs_bp_access_handler~save_additional_object_data.

  FIELD-SYMBOLS:
        <lt_data_new>     TYPE mdg_bs_bp_s_ecc_extern,
        <lt_data_db>      TYPE mdg_bs_bp_s_ecc_extern,
        <ls_partner>      TYPE bus_ei_extern,
        <ls_data>         TYPE zsecret_number,
        <ls_data_db>      TYPE zsecret_number,
        <ls_company>      TYPE vmds_ei_company,
        <ls_secret>       TYPE zei_secret_number,
        <lv_vendor>       TYPE mdg_bs_bp_s_mlt_as_suppl,
        <ls_data_ext>     TYPE mdg_bs_bp_s_ecc_extern.



  DATA: lt_current_secret TYPE TABLE OF zsecret_number,
        lt_database_secret TYPE TABLE OF zsecret_number,
        ls_current_secret TYPE zsecret_number,
        ls_database_secret TYPE zsecret_number,
        lt_ins             TYPE TABLE OF zsecret_number,
        lt_upd             TYPE TABLE OF zsecret_number,
        lt_del             TYPE TABLE OF zsecret_number,
        ls_data            TYPE zsecret_number,
```

```
        lr_ecc_extern_db    TYPE REF TO data.

* Note: IS_DATA_DB does not contain DB state of customer specific data, use GT_ECC_EXTERN_DB for current DB state of handler
  ASSIGN is_data TO <lt_data_new>.

  ASSIGN COMPONENT 'PARTNER' OF STRUCTURE is_data TO <ls_partner>.
  CHECK <ls_partner> IS ASSIGNED.

  READ TABLE gt_ecc_extern_db
        ASSIGNING <ls_data_ext>
         WITH KEY partner-header-object_instance-bpartner = <ls_partner>-header-object_instance-bpartner.


  IF <ls_data_ext> IS ASSIGNED.
    ASSIGN <ls_data_ext> TO <lt_data_db>.
  ELSE.
    CREATE DATA lr_ecc_extern_db TYPE mdg_bs_bp_s_ecc_extern.
    ASSIGN lr_ecc_extern_db->* TO <lt_data_db>.

  ENDIF.

* current data
  LOOP AT <lt_data_new>-vendors ASSIGNING <lv_vendor>.
    LOOP AT <lv_vendor>-vendor-company_data-company ASSIGNING <ls_company>.
      LOOP AT <ls_company>-zzsecret_number-zzsecret_number ASSIGNING <ls_secret>.
        CLEAR: ls_data.
        ASSERT NOT <lv_vendor>-vendor-header-object_instance-lifnr IS INITIAL.
        ls_data-lifnr         = <lv_vendor>-vendor-header-object_instance-lifnr.
        ls_data-bukrs         = <ls_company>-data_key-bukrs.
        ls_data-secret_number = <ls_secret>-data_key-secret_number.
        ls_data-sn_comment    = <ls_secret>-data-sn_comment.
        CASE  <ls_secret>-task.
          WHEN gc_del.
            APPEND ls_data  TO lt_del.
          WHEN gc_ins.
```

```
            APPEND  ls_data TO lt_ins.
          WHEN gc_upd.
            APPEND ls_data TO lt_upd.
        ENDCASE
      .ENDLOOP.
    ENDLOOP
  .ENDLOOP.


* database data
  LOOP AT <lt_data_db>-vendors ASSIGNING <lv_vendor>.
    LOOP AT <lv_vendor>-vendor-company_data-company ASSIGNING <ls_company>.
      LOOP AT <ls_company>-zzsecret_number-
zzsecret_number ASSIGNING <ls_secret>.
        ASSERT NOT <lv_vendor>-vendor-header-object_instance-lifnr IS INITIAL.
        ls_database_secret-lifnr         = <lv_vendor>-vendor-header-
object_instance-lifnr.
        ls_database_secret-bukrs         = <ls_company>-data_key-bukrs.
        ls_database_secret-secret_number = <ls_secret>-data_key-secret_number.
        ls_database_secret-sn_comment    = <ls_secret>-data-sn_comment.
        APPEND ls_database_secret TO lt_database_secret.
      ENDLOOP
    .ENDLOOP.
  ENDLOOP.


  lt_current_secret = lt_database_secret.


* merge
* Deletes
  LOOP AT lt_del ASSIGNING <ls_data>.
    READ TABLE lt_current_secret  WITH KEY bukrs         =  <ls_data>-bukrs
                                           lifnr         =  <ls_data>-lifnr
                                           secret_number = <ls_data>-
secret_number
                                    ASSIGNING <ls_data_db>.
    IF sy-subrc = 0.

*       delete record in target table
      DELETE lt_current_secret INDEX sy-tabix.
```

```
    ELSE.
*       record doesn't exist on DB -
> must be a new one, keep data to be checked consistent
      READ TABLE lt_ins  WITH KEY bukrs =  <ls_data>-bukrs
                                           lifnr = <ls_data>-lifnr
                                           secret_number = <ls_data>-
secret_number
                          ASSIGNING <ls_data_db>.
      IF sy-subrc = 0.

        DELETE lt_current_secret INDEX sy-tabix.
      ENDIF.
    ENDIF.

  ENDLOOP.


* Inserts
  LOOP AT lt_ins ASSIGNING <ls_data>.
    READ TABLE lt_current_secret  WITH KEY bukrs =  <ls_data>-bukrs
                                           lifnr = <ls_data>-lifnr
                                           secret_number = <ls_data>-
secret_number
                                  ASSIGNING <ls_data_db>.
*     insert record into target table
    IF sy-subrc NE 0.
      INSERT  <ls_data> INTO TABLE lt_current_secret.
   ENDIF.
 ENDLOOP.


* Updates
  LOOP AT lt_upd ASSIGNING <ls_data>.
    READ TABLE lt_current_secret  WITH KEY bukrs =  <ls_data>-bukrs
                                           lifnr = <ls_data>-lifnr
                                           secret_number = <ls_data>-
secret_number
                                  ASSIGNING <ls_data_db>.
*     insert record into target table
    IF sy-subrc NE 0.
      <ls_data_db> = <ls_data>.
```

```
    ENDIF.
  ENDLOOP.

  CALL FUNCTION 'Z_SECRET_NUMBER_UPDATES' IN UPDATE TASK
    TABLES
      x_secret_numbers = lt_current_secret
      y_secret_numbers = lt_database_secret.

ENDMETHOD.
```

SORT_ENTITIES

```
method
IF_MDG_BS_BP_ACCESS_HANDLER~SORT_ENTITIES.

                    DATA: lt_entity TYPE TABLE OF ty_usmd_entity,
                          lv_entity TYPE ty_usmd_entity.

* supplier entities are appended to table ct_entities which already at least contains BP entities sorted by
* BP handler before



  lt_entity  = it_entities_all.

  READ TABLE lt_entity
  WITH KEY entity = 'ZSECRET_N'
  INTO lv_entity. IF sy-
  subrc = 0.
    APPEND lv_entity TO ct_entities.
  ENDIF.

endmethod.
```

5.1.2 Helper Methods

CONSTRUCTOR

```
method CONSTRUCTOR.
  DATA:
      ls_bp_fldgrp           TYPE MDG_BS_BP_FLDGRP,
      ls_suppl_fldgrp        TYPE MDG_BS_SUPPL_FG,
      ls_usmd_entity         TYPE usmd_entity,
      lt_bp_fldgrp           TYPE table of MDG_BS_BP_FLDGRP,
      gt_bp_fldgrp           TYPE table of MDG_BS_BP_FLDGRP,
      ls_strucname           TYPE BU_BAPISTRC,
      ls_MDG_ECC_BPFLDMAP    TYPE MDG_ECC_BPFLDMAP,
      lt_MDG_ECC_BPFLDMAP    TYPE table of MDG_ECC_BPFLDMAP,
      gt_MDG_ECC_BPFLDMAP    TYPE table of MDG_ECC_BPFLDMAP,
      ls_field_mapping_ecc   TYPE USMD_S_MAP_STRUC,
      lt_field_mapping_ecc   TYPE USMD_TS_MAP_STRUC,
      lt_strucnames_ecc      TYPE MDG_BS_BP_STRUCNAMES_T,
      lt_usmd_entity         TYPE table of usmd_entity.

  CALL METHOD super->constructor.

  me->gv_wf_services = cl_mdg_bs_suppl_wf_services=>get_instance( ).

* 1. get all included entities
  select usmd_entity from usmd0022 into  ls_usmd_entity
                                   where usmd_model = 'BP'
                                   and   USMD_OBJSTAT = 'A'.
    collect ls_usmd_entity into lt_usmd_entity.
  endselect.

* 2. get mapping information for each entity
  loop at lt_usmd_entity into ls_usmd_entity.
    refresh lt_bp_fldgrp.
    refresh lt_strucnames_ecc.
    refresh lt_field_mapping_ecc.
    CALL METHOD
      CL_MDG_BS_FND_BP_SERVICES=>GET_FIELDMAPPING_FOR_ENTITY
      EXPORTING
*       IO_MODEL             =
        IV_ENTITY            = ls_usmd_entity
      IMPORTING
        et_strucnames_ecc    = lt_strucnames_ecc
```

```
        et_field_mapping_ecc = lt_field_mapping_ecc
        ET_BP_FLDGRP         = lt_bp_fldgrp.
    append lines of lt_bp_fldgrp to gt_bp_fldgrp.

* map ecc structures
    loop at lt_strucnames_ecc into ls_strucname.
      refresh lt_mdg_ecc_bpfldmap.
      SELECT * FROM mdg_ecc_bpfldmap into TABLE lt_mdg_ecc_bpfldmap
                                      WHERE bapistrucname    = ls_strucname.
      check sy-subrc = 0.
      append lines of lt_mdg_ecc_bpfldmap to gt_mdg_ecc_bpfldmap.
    endloop.
    loop at lt_field_mapping_ecc into ls_field_mapping_ecc.
      READ TABLE gt_mdg_ecc_bpfldmap WITH KEY bapistrucname = ls_field_mapping_e
cc-tab_source
                                              bapifldnm     = ls_field_mapping_ecc-
fld_source INTO ls_mdg_ecc_bpfldmap.
      check sy-subrc = 0.
      ls_suppl_fldgrp-model     = 'BP'.
      ls_suppl_fldgrp-entity    = ls_usmd_entity.
      ls_suppl_fldgrp-attribute = ls_field_mapping_ecc-fieldname.
      ls_suppl_fldgrp-OBJECT_TYPE_CODE  = '147'.
      ls_suppl_fldgrp-modif   =   ls_mdg_ecc_bpfldmap-modif.
      ls_suppl_fldgrp-tabnm   =   ls_mdg_ecc_bpfldmap-tabnm.
      ls_suppl_fldgrp-feldn   =   ls_mdg_ecc_bpfldmap-feldn.
      APPEND ls_suppl_fldgrp TO gt_suppl_fldgrp.
    endloop.
  endloop.

* 3. map BP entities to gt_suppl_fldgrp
  loop at gt_bp_fldgrp into ls_bp_fldgrp.
    move-corresponding ls_bp_fldgrp to ls_suppl_fldgrp.
    ls_suppl_fldgrp-feldn = ls_bp_fldgrp-feldn.
    ls_suppl_fldgrp-tabnm = ls_bp_fldgrp-tabnm.
    ls_suppl_fldgrp-model = 'BP'.
    ls_suppl_fldgrp-object_type_code = '147'.
    collect ls_suppl_fldgrp into gt_suppl_fldgrp.
  endloop.
  delete adjacent duplicates from gt_bp_fldgrp.
```

```
  delete adjacent duplicates from gt_suppl_fldgrp.


* get instance of multiple assignment memory
  go_mlt_as_api = cl_mdg_bs_bp_mlt_assgnmnt_api=>get_instance( ).

endmethod.
```

MAP_BP_ZSECRET_N_2API

```
method MAP_BP_ZSECRET_N_2API.

  DATA: lv_dummy         TYPE          string,                "#EC NEEDED
        ls_secret        TYPE          ZCT_S_BP_PP_BP_SECRET,
        ls_dunn_x        TYPE          mdg_bs_suppl_bp_dunn_x,
        ls_target        TYPE          ZEI_SECRET_NUMBER,
        ls_comp          TYPE          vmds_ei_company,
        ls_comp_db       TYPE          vmds_ei_company.


  FIELD-SYMBOLS:
    <ls_data_x>           TYPE any,
    <ls_comp>             TYPE vmds_ei_company,
    <lv_assgnm_id>        TYPE mdg_bp_assignment_id,
    <lv_partner>          TYPE bu_partner,
    <ls_ecc_extern_db>    TYPE mdg_bs_bp_s_ecc_extern,
    <ls_cvis_ei_extern>   TYPE cvis_ei_extern,
    <ls_vendors>          TYPE mdg_bs_bp_s_mlt_as_suppl,
    <ls_vendor>           TYPE vmds_ei_extern.

  MOVE-CORRESPONDING is_data TO ls_secret.

* Company Data key
  READ TABLE cs_suppl-vendor-company_data-company WITH KEY data_key-
bukrs = ls_secret-company
                                                    ASSIGNING <ls_comp>.
  IF <ls_comp> IS NOT ASSIGNED.
```

```
* create entry as entities BP_COMPNY and BP_DUNN may map to the same target record as dunning data
are
* company dependent
* CAUTION if no inactive changes for the company code exist: check if company data for the relevant
*comp. code is saved. If DB data exists, the company code data must be insertedas well as otherwise
* check would determine initial company code data when checking entity BP_COMPNY
    ls_comp-data_key-bukrs = ls_secret-company.
    ls_comp-
task = gc_upd. "if insert of entity comes later on task is overwritten

*       check if comp. code is saved
    ASSIGN COMPONENT 'BP_HEADER' OF STRUCTURE is_data TO <lv_partner>.
    IF sy-subrc = 0.
      IF <lv_partner> IS NOT INITIAL.

        ASSIGN COMPONENT 'ASSGNM_ID' OF STRUCTURE is_data TO <lv_assgnm_id>.
        IF sy-subrc = 0.
          IF <lv_assgnm_id> IS NOT INITIAL.
            READ TABLE gt_ecc_extern_db ASSIGNING <ls_ecc_extern_db>
                                        WITH KEY partner-header-object_instance-
bpartner = <lv_partner>.
            IF sy-subrc = 0.
              READ TABLE <ls_ecc_extern_db>-vendors ASSIGNING <ls_vendors>
                                        WITH KEY assignment_id = <lv_assgnm_id>.
              IF sy-subrc = 0.
                READ TABLE <ls_vendors>-vendor-company_data-
company INTO ls_comp_db
                                        WITH KEY data_key-bukrs = ls_secret-
company.
                IF sy-subrc = 0.

                  ls_comp-data = ls_comp_db-data.
                ENDIF.
              ENDIF.

            ENDIF.
          ENDIF.

        ENDIF.
      ENDIF.
```

```
    ENDIF.
    INSERT ls_comp INTO TABLE cs_suppl-vendor-company_data-
company ASSIGNING <ls_comp>.
  ENDIF.


* Secret Number Data Key
  ls_target-data_key-SECRET_NUMBER = ls_secret-ZSECRET.

* Secret Number Data
  IF iv_task = gc_ins OR iv_task = gc_upd.
    ls_target-data-SN_COMMENT = ls_secret-SNCOMMENT .
  ENDIF.

  ls_target-task = iv_task.
  APPEND ls_target TO <ls_comp>-ZZSECRET_NUMBER-ZZSECRET_NUMBER.

  cv_xchange = abap_true.

endmethod.
```

MAP_BP_ZSECRET_N_2STA

```
method MAP_BP_ZSECRET_N_2STA.

  FIELD-SYMBOLS: <ls_extern_db>          like line of GT_ECC_EXTERN_DB,
                 <ls_vendor>             type MDG_BS_BP_S_MLT_AS_SUPPL,
                 <ls_company>            type line of VMDS_EI_COMPANY_T,
                 <ls_secret_number>      type ZEI_SECRET_NUMBER,
                 <lv_bp_header>          type BU_PARTNER,
                 <lv_company>            type bukrs,
                 <lv_secret_number>      type ZZSECRET_NUMBER,
                 <lv_comment>            type ZZSECRET_NUMBER_COMMENT.

  DATA: lr_data       TYPE REF TO   data,
        lv_dummy      TYPE          string.
```

```
  FIELD-SYMBOLS:
        <ls_data>    TYPE data,
        <ls_comp>    TYPE vmds_ei_company,
        <ls_secret>  TYPE ZEI_SECRET_NUMBER,
        <lv_as_id>   TYPE mdg_bp_assignment_id.


  loop at GT_ECC_EXTERN_DB assigning <ls_extern_db>.
    loop at <ls_extern_db>-vendors assigning <ls_vendor>.
      loop at <ls_vendor>-VENDOR-COMPANY_DATA-COMPANY assigning <ls_company>.
        loop at <ls_company>-ZZSECRET_NUMBER-
ZZSECRET_NUMBER assigning <ls_secret>.

          CREATE DATA lr_data LIKE LINE OF ct_data.
          ASSIGN lr_data->*              TO <ls_data>.

          ASSIGN COMPONENT:
              'BP_HEADER' OF STRUCTURE <ls_data> TO <lv_bp_header>,
              'COMPANY'   OF STRUCTURE <ls_data> TO <lv_company>,
              'ZSECRET'   OF STRUCTURE <ls_data> TO <lv_secret_number>,
              'SNCOMMENT' OF STRUCTURE <ls_data> TO <lv_comment>.

          <lv_bp_header>     = <ls_extern_db>-partner-header-object_instance-
bpartner.
          <lv_company>       = <ls_company>-data_key-bukrs.
          <lv_secret_number> = <ls_secret>-data_key-SECRET_NUMBER.
          <lv_comment>       = <ls_secret>-data-SN_COMMENT.

          IF iv_assignment_id IS SUPPLIED.
            ASSIGN COMPONENT 'ASSGNM_ID' OF STRUCTURE <ls_data> TO <lv_as_id>.
            IF sy-subrc = 0.
              <lv_as_id> = iv_assignment_id.
            ENDIF.
          ENDIF.

          INSERT <ls_data>             INTO TABLE ct_data.

        endloop.
      ENDLOOP.
```

```
    ENDLOOP.
  ENDLOOP.


endmethod.
```

READ_VENDOR

```
method READ_VENDOR.

     DATA:
      lo_ka_bp_vendor      TYPE REF TO cvi_ka_bp_vendor,
      ls_bp_suppl_link     TYPE ty_bp_suppl_link,
      ls_vmds_ei_main      TYPE vmds_ei_main,
      ls_vmds_ei_extern    TYPE vmds_ei_extern,
      ls_vmds_ei_main_in   TYPE vmds_ei_main,
      ls_cvis_extern       TYPE cvis_ei_extern,
      lv_partner_guid      TYPE bu_partner_guid.


    FIELD-SYMBOLS:
      <ls_idlist>          TYPE bus_ei_instance.

* Read Connection Between Vendor and Business Partner
    lo_ka_bp_vendor = cvi_ka_bp_vendor=>get_instance( ).

    LOOP AT it_idlist ASSIGNING <ls_idlist>.
* existenz in globaler Tabelle noch prüfen ls_bp_suppl_link-bpartner =
      <ls_idlist>-bpartner. lv_partner_guid   = <ls_idlist>-bpartnerguid.

      IF lv_partner_guid IS INITIAL.
        CALL FUNCTION 'BUPA_NUMBERS_GET'
          EXPORTING
            iv_partner      = ls_bp_suppl_link-bpartner
          IMPORTING
            ev_partner_guid = lv_partner_guid.
      ENDIF.
```

```
      lo_ka_bp_vendor->get_assigned_vendor_for_bp(
        EXPORTING
          i_partner = lv_partner_guid
        RECEIVING
          r_vendor = ls_bp_suppl_link-supplier
        EXCEPTIONS

          OTHERS     = 1 ).

      IF sy-subrc = 0 AND ls_bp_suppl_link-supplier IS NOT INITIAL.
        APPEND ls_bp_suppl_link TO gt_bp_suppl_link.
        ls_vmds_ei_extern-header-object_task = 'M'.
        ls_vmds_ei_extern-header-object_instance-lifnr = ls_bp_suppl_link-
supplier.
        APPEND ls_vmds_ei_extern TO ls_vmds_ei_main_in-vendors.
        CLEAR: ls_vmds_ei_extern.

      ENDIF.

    ENDLOOP.
    IF NOT ls_vmds_ei_main_in IS INITIAL.
* Extract Vendors
      vmd_ei_api_extract=>get_data(
        EXPORTING
          is_master_data = ls_vmds_ei_main_in
        IMPORTING
          es_master_data = ls_vmds_ei_main
        EXCEPTIONS
          OTHERS         = 1 ).
      LOOP AT it_idlist ASSIGNING <ls_idlist>.
        ls_cvis_extern-partner-header-object_instance-bpartner = <ls_idlist>-
bpartner.
        CLEAR ls_cvis_extern-vendor.
*    Vendor
        READ TABLE gt_bp_suppl_link
              INTO ls_bp_suppl_link
          WITH KEY bpartner = <ls_idlist>-bpartner.
        IF sy-subrc = 0.
```

```
          READ TABLE ls_vmds_ei_main-vendors
                INTO ls_vmds_ei_extern
            WITH KEY header-object_instance-lifnr = ls_bp_suppl_link-supplier.
          IF sy-subrc = 0.
            ls_cvis_extern-vendor = ls_vmds_ei_extern.
          ENDIF.
        ENDIF.
        APPEND ls_cvis_extern TO et_supplier.
      ENDLOOP.

    ENDIF.

endmethod.
```

READ_VENDOR_BY_LIFNR

```
method READ_VENDOR_BY_LIFNR.

  DATA:
      ls_vmds_ei_main          TYPE vmds_ei_main,
      ls_vmds_ei_extern        TYPE vmds_ei_extern,
      ls_vmds_ei_main_in       TYPE vmds_ei_main,
      ls_cvis_extern           TYPE cvis_ei_extern,
      lv_partner_guid          TYPE bu_partner_guid,


  FIELD-SYMBOLS:
    <ls_idlist>                TYPE bus_ei_instance.


  IF iv_lifnr IS NOT INITIAL. ls_vmds_ei_extern-header-
    object_task = 'M'.
    ls_vmds_ei_extern-header-object_instance-lifnr = iv_lifnr.APPEND
    ls_vmds_ei_extern TO ls_vmds_ei_main_in-vendors.
  ENDIF.

  IF NOT ls_vmds_ei_main_in IS INITIAL.
```

```
* Extract Vendors
    vmd_ei_api_extract=>get_data(
      EXPORTING
        is_master_data = ls_vmds_ei_main_in
      IMPORTING
        es_master_data = ls_vmds_ei_main
      EXCEPTIONS
        OTHERS         = 1 ).

    LOOP AT ls_vmds_ei_main-vendors INTO ls_vendors. "only one entry expected
      es_supplier-vendor = ls_vendors.
      EXIT.
    ENDLOOP.
  ENDIF.

endmethod.
```

5.1.3 Function Modules

Z_SECRET_NUMBER_UPDATES

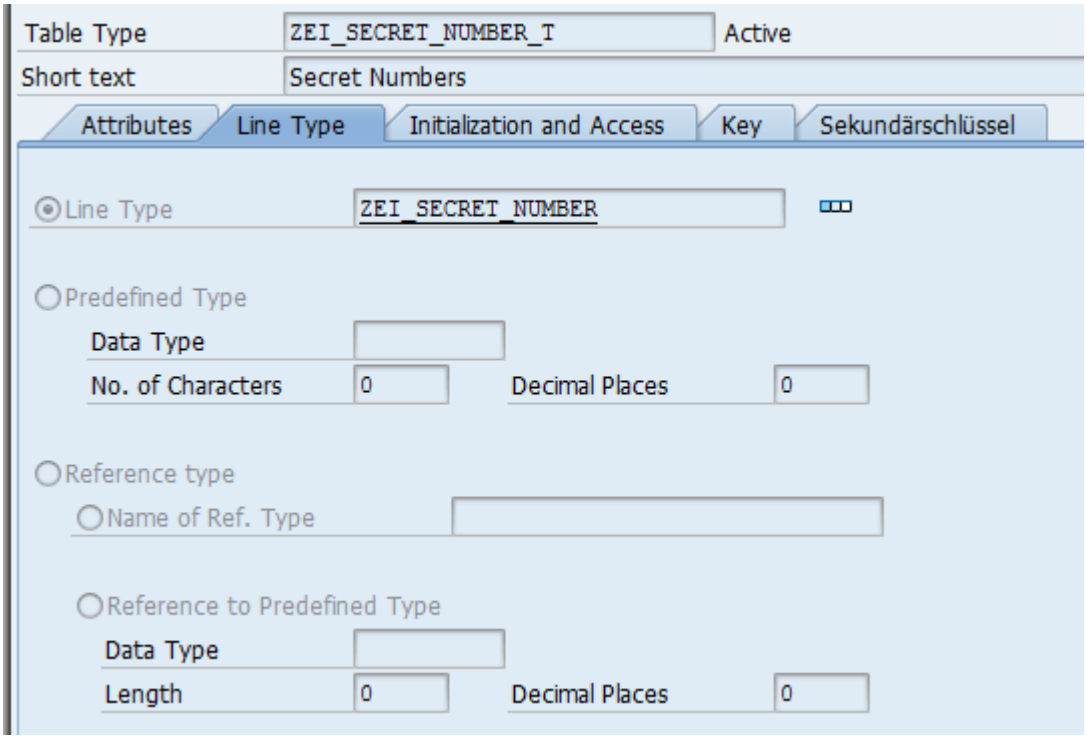
```
FUNCTION Z_SECRET_NUMBER_UPDATES.

*"----------------------------------------------------------------------------------------------
*"*"Update Function Module:
*"
*"*"Local Interface:
*"  TABLES
*"      X_SECRET_NUMBERS STRUCTURE  ZSECRET_NUMBER
*"      Y_SECRET_NUMBERS STRUCTURE  ZSECRET_NUMBER
*"
*"----------------------------------------------------------------------------------------------

  data: ls_y_secret_numbers type ZSECRET_NUMBER.

  loop at Y_SECRET_NUMBERS INTO ls_y_secret_numbers.
    read table X_SECRET_NUMBERS with key BUKRS         = ls_y_secret_numbers-
bukrs
```

```
                                                LIFNR                = ls_y_secret_numbers-
lifnr
                                                SECRET_NUMBER = ls_y_secret_numbers-
secret_number.
      if sy-subrc <> 0.
        delete from ZSECRET_NUMBER where BUKRS                = ls_y_secret_numbers-bukrs
                                            and      LIFNR                = ls_y_secret_numbers-lifnr
                                          and      SECRET_NUMBER = ls_y_secret_numbers-
secret_number.
        if sy-subrc <> 0.
          message x000(ZSECRET_NUMBERS).
        endif.

      endif.
  ENDLOOP.

  modify ZSECRET_NUMBER from table X_SECRET_NUMBERS. "Current State
  if sy-subrc <> 0.
    message x000(ZSECRET_NUMBERS).


ENDFUNCTION.
```

5.2 Data Dictionary Objects for Extension of VMDS_EI_VMD_COMPANY

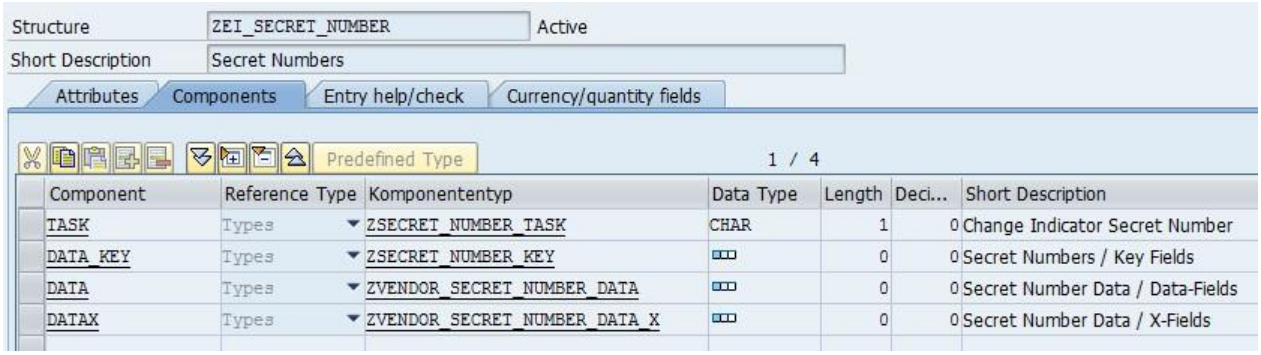
5.2.1 Table Types

ZEI_SECRET_NUMBER_T



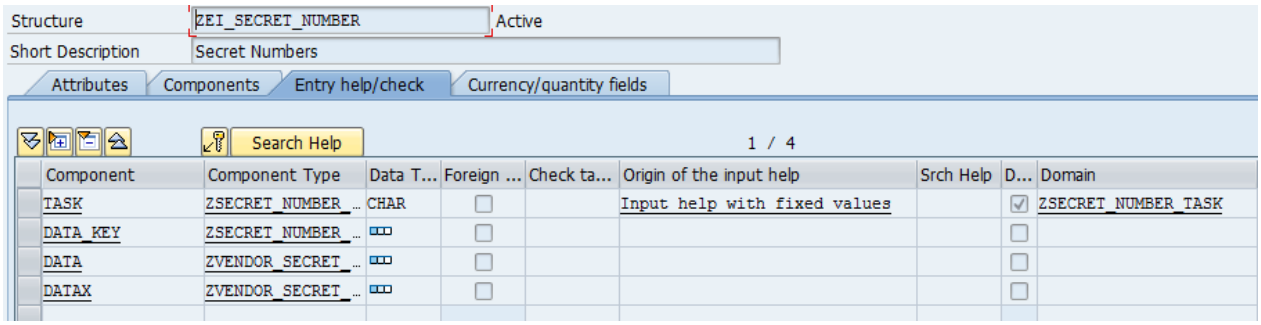
5.2.2 Structures

ZEI_SECRET_NUMBER



Structure ZEI_SECRET_NUMBER Active
Short Description Secret Numbers
Attributes | Components | Entry help/check | Currency/quantity fields
Predefined Type 1 / 4

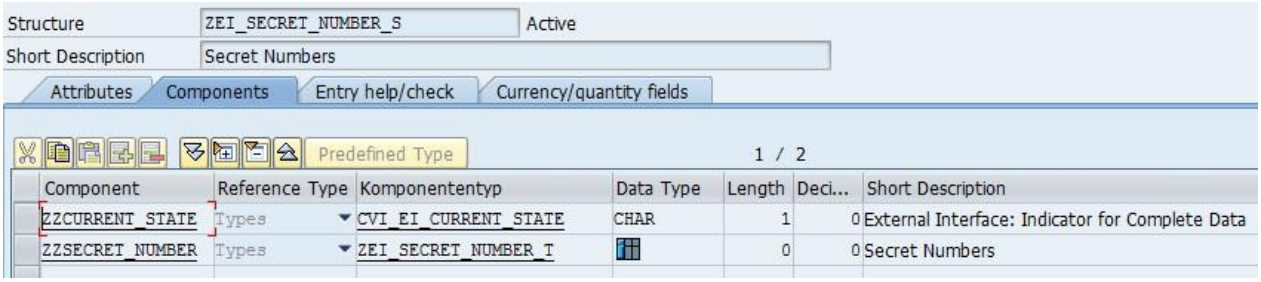
Component	Reference Type	Komponententyp	Data Type	Length	Deci...	Short Description
TASK	Types	ZSECRET_NUMBER_TASK	CHAR	1	0	Change Indicator Secret Number
DATA_KEY	Types	ZSECRET_NUMBER_KEY		0	0	Secret Numbers / Key Fields
DATA	Types	ZVENDOR_SECRET_NUMBER_DATA		0	0	Secret Number Data / Data-Fields
DATAX	Types	ZVENDOR_SECRET_NUMBER_DATA_X		0	0	Secret Number Data / X-Fields



Structure ZEI_SECRET_NUMBER Active
Short Description Secret Numbers
Attributes | Components | Entry help/check | Currency/quantity fields
Search Help 1 / 4

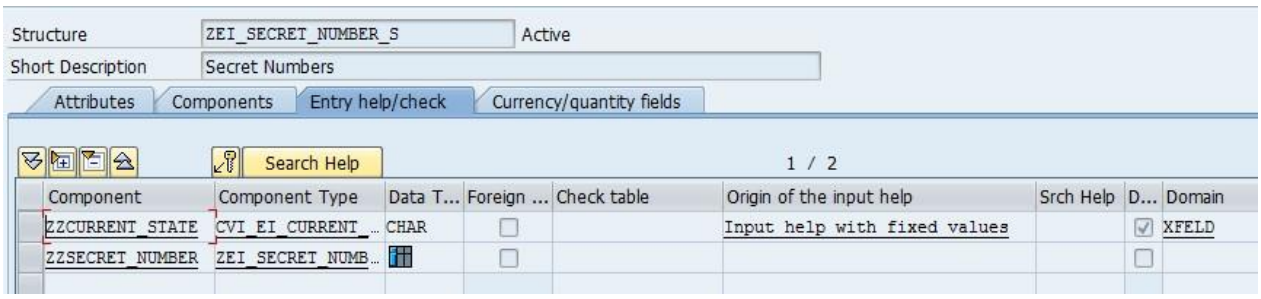
Component	Component Type	Data T...	Foreign ...	Check ta...	Origin of the input help	Srch Help	D...	Domain
TASK	ZSECRET_NUMBER_...	CHAR	☐		Input help with fixed values		☑	ZSECRET_NUMBER_TASK
DATA_KEY	ZSECRET_NUMBER_...		☐				☐	
DATA	ZVENDOR_SECRET_...		☐				☐	
DATAX	ZVENDOR_SECRET_...		☐				☐	

ZEI_SECRET_NUMBER_S



Structure ZEI_SECRET_NUMBER_S Active
Short Description Secret Numbers
Attributes | Components | Entry help/check | Currency/quantity fields
Predefined Type 1 / 2

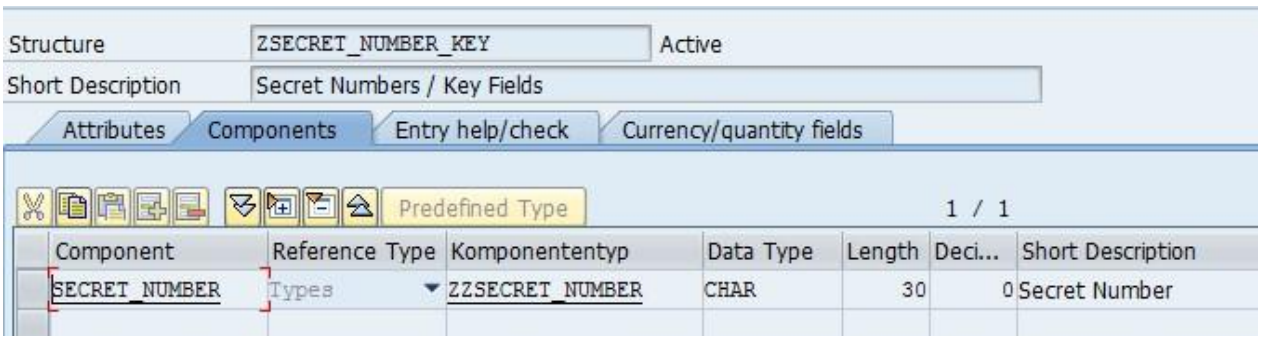
Component	Reference Type	Komponententyp	Data Type	Length	Deci...	Short Description
ZZCURRENT_STATE	Types	CVI_EI_CURRENT_STATE	CHAR	1	0	External Interface: Indicator for Complete Data
ZZSECRET_NUMBER	Types	ZEI_SECRET_NUMBER_T		0	0	Secret Numbers

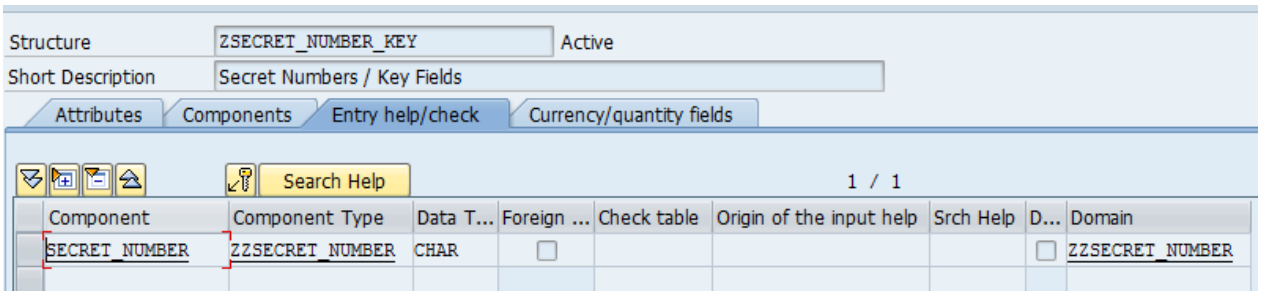


Structure ZEI_SECRET_NUMBER_S Active
Short Description Secret Numbers
Attributes | Components | Entry help/check | Currency/quantity fields
Search Help 1 / 2

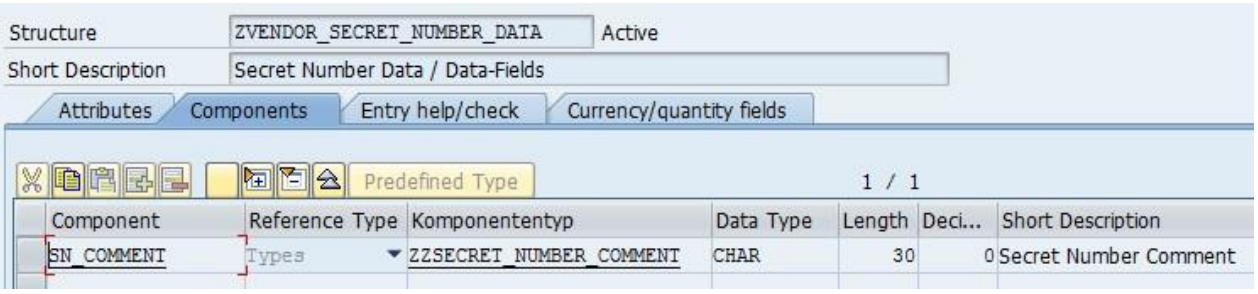
Component	Component Type	Data T...	Foreign ...	Check table	Origin of the input help	Srch Help	D...	Domain
ZZCURRENT_STATE	CVI_EI_CURRENT_...	CHAR	☐		Input help with fixed values		☑	XFELD
ZZSECRET_NUMBER	ZEI_SECRET_NUMB...		☐				☐	

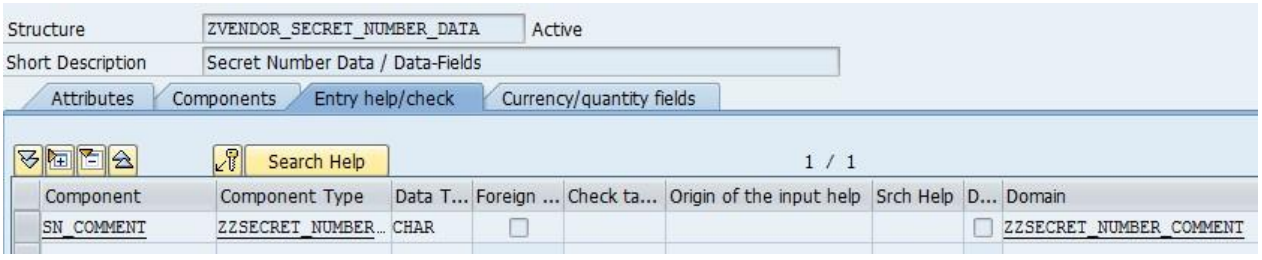
ZSECRET_NUMBER_KEY



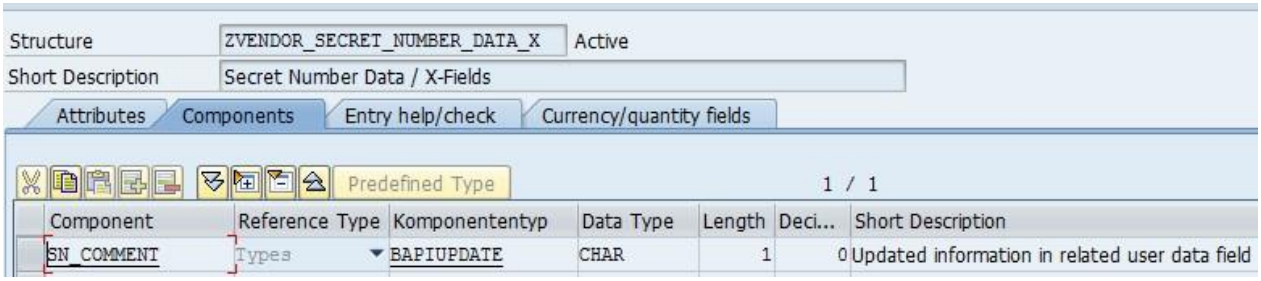


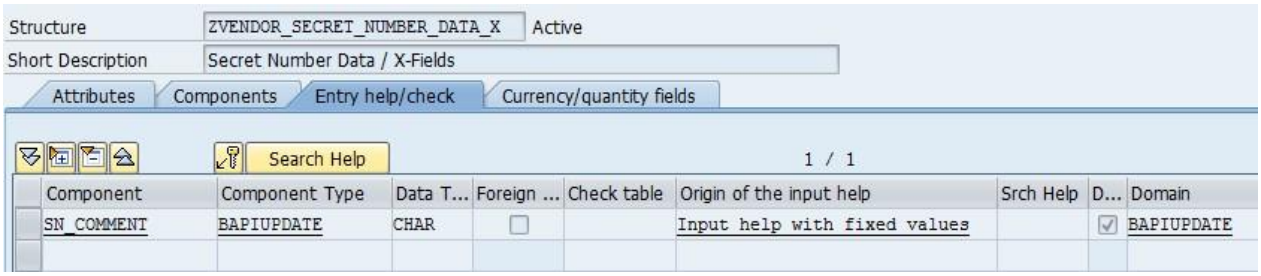
ZVENDOR_SECRET_NUMBER_DATA



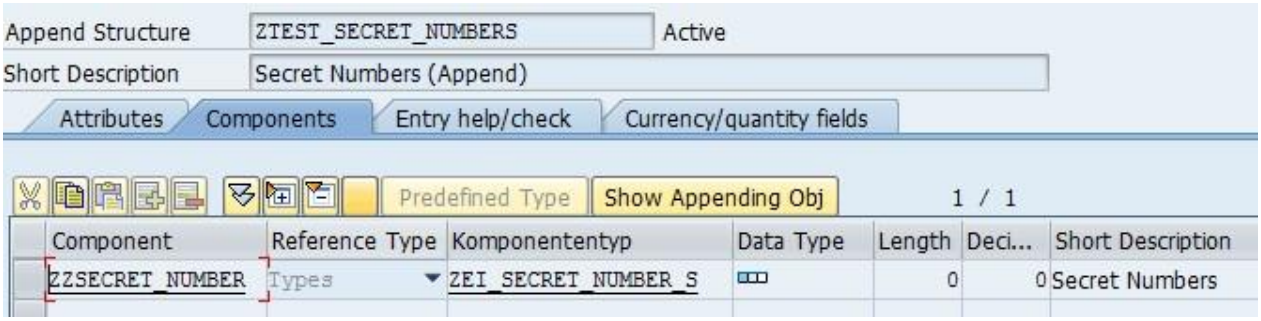


ZVENDOR_SECRET_NUMBER_DATA_X



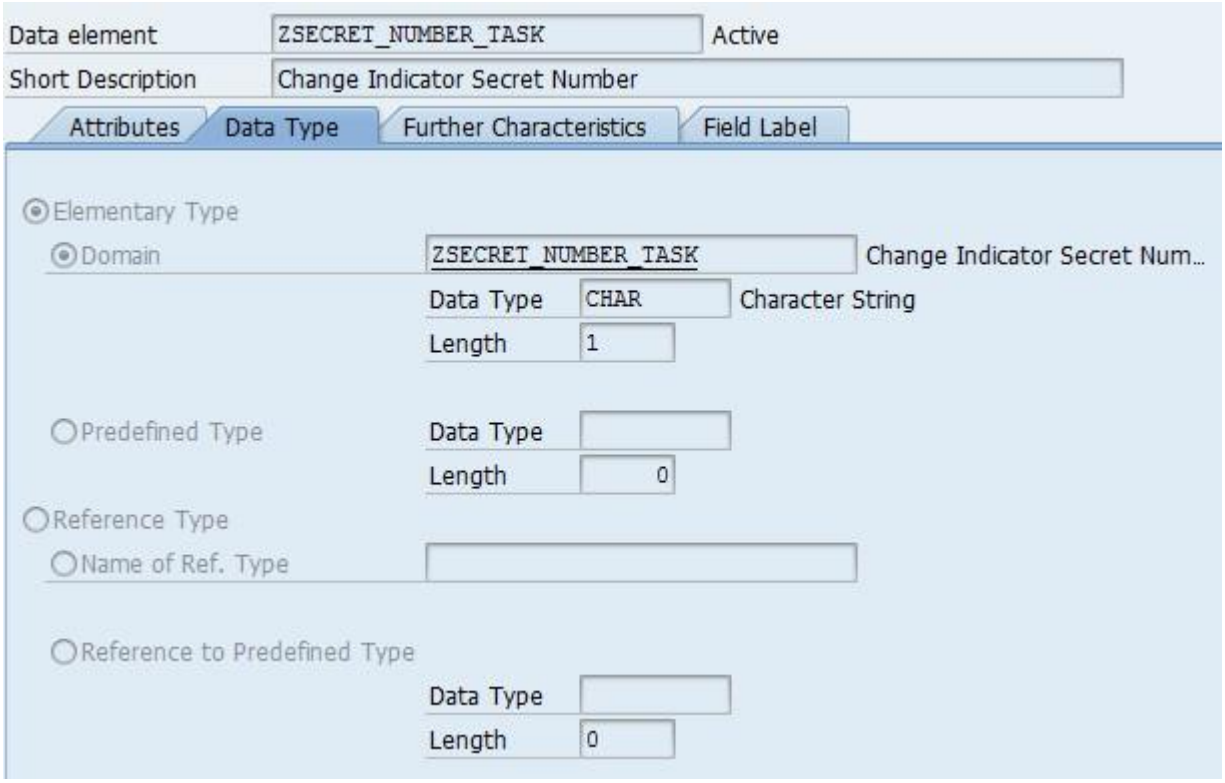


ZTEST_SECRET_NUMBERS

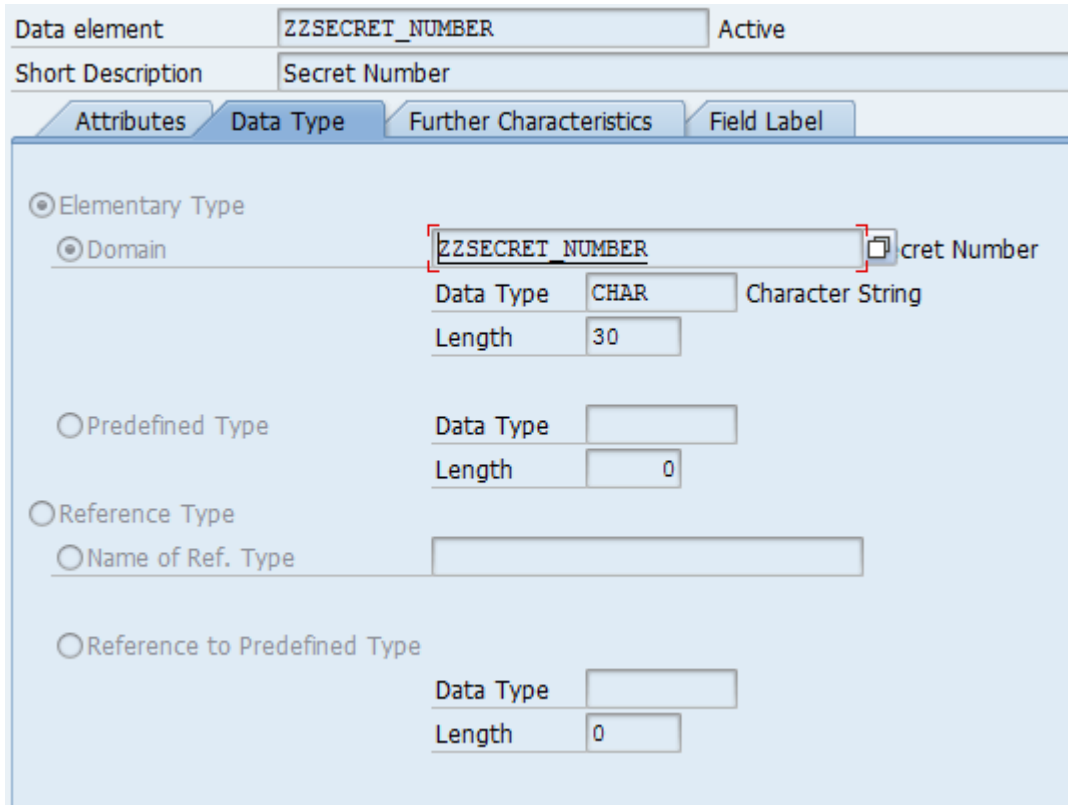


5.2.3 Data Elements

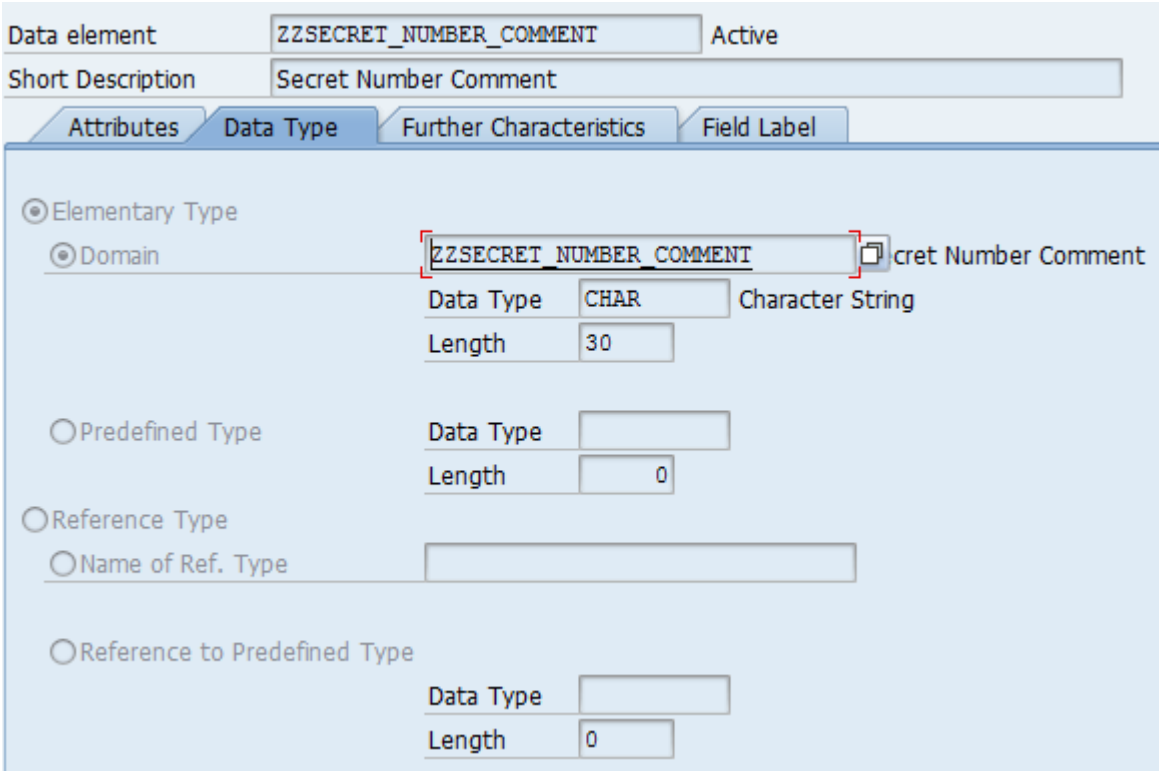
ZSECRET_NUMBER_TASK



ZZSECRET_NUMBER

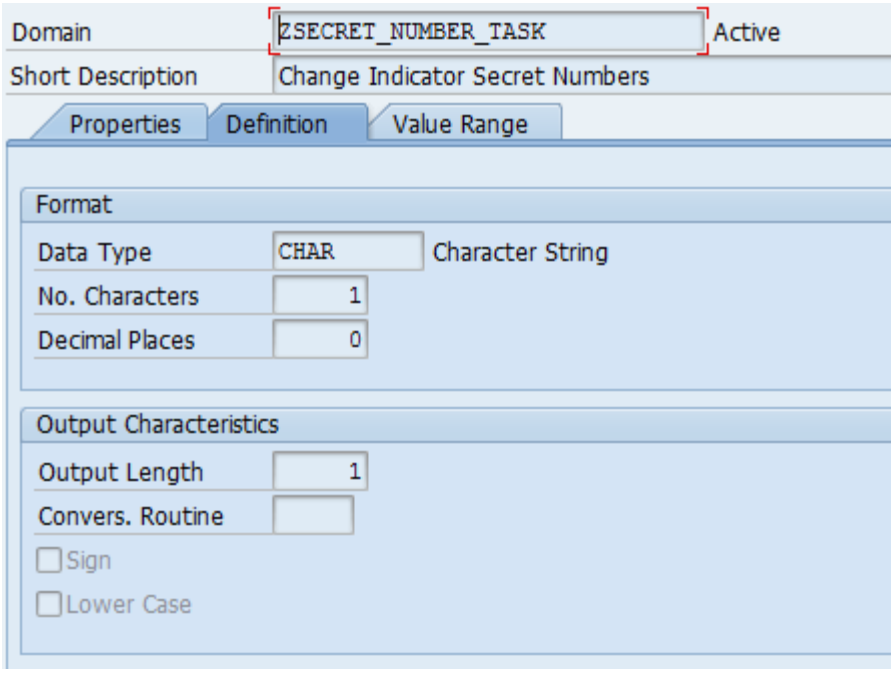


ZZSECRET_NUMBER_COMMENT

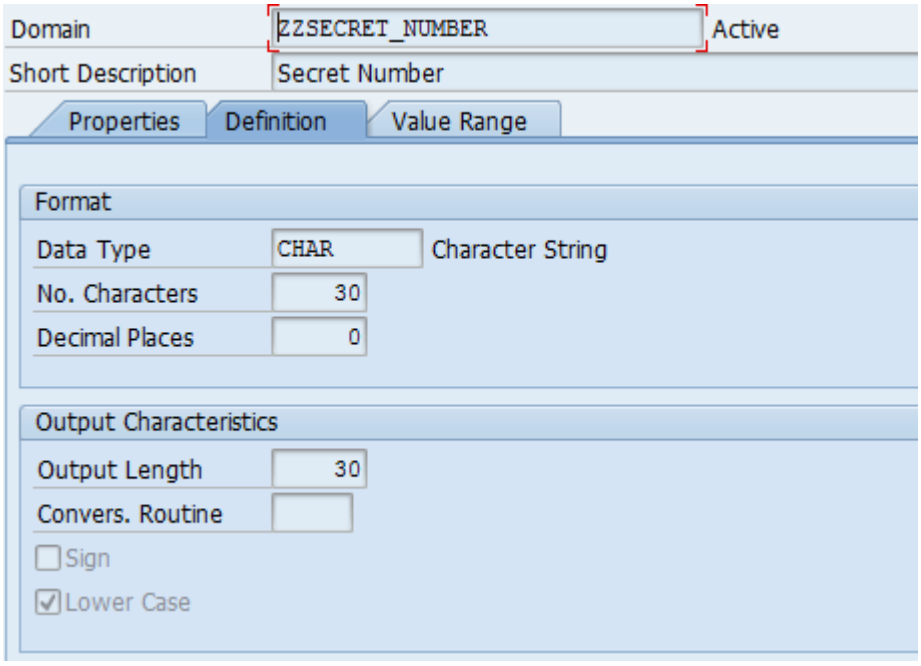


5.2.4 Domains

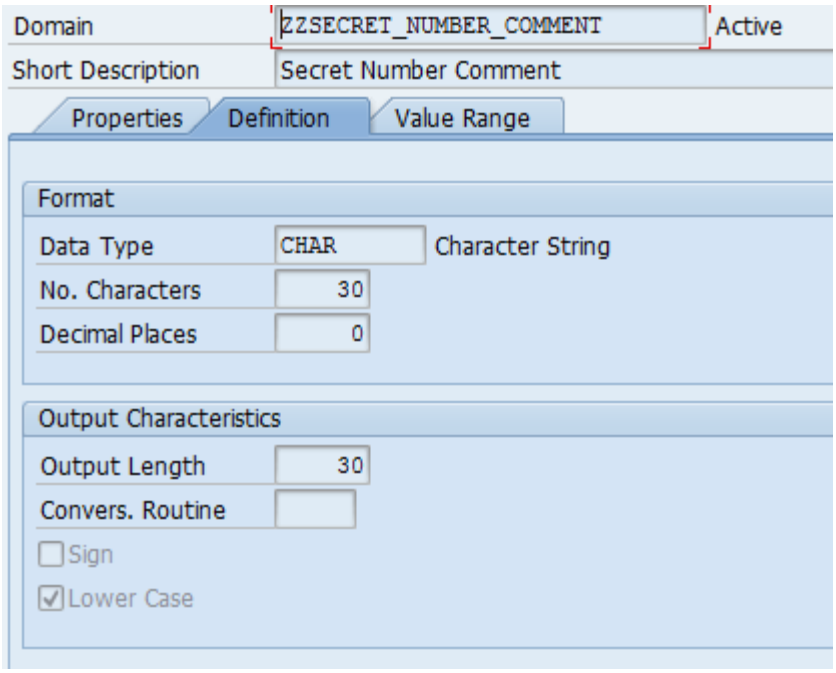
ZSECRET_NUMBER_TASK



ZZSECRET_NUMBER



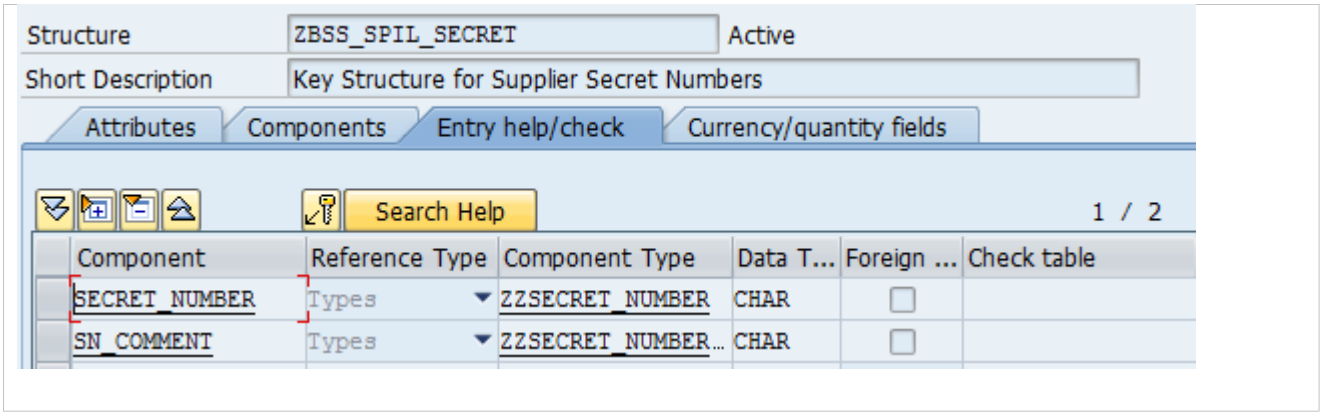
ZZSECRET_NUMBER_COMMENT



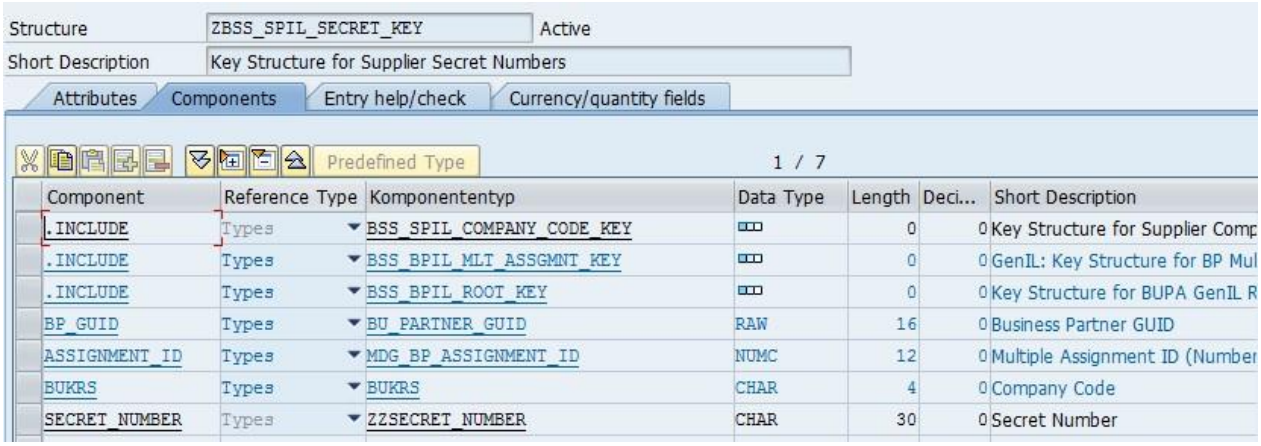
5.3 Data Dictionary Objects for genIL Model Extension

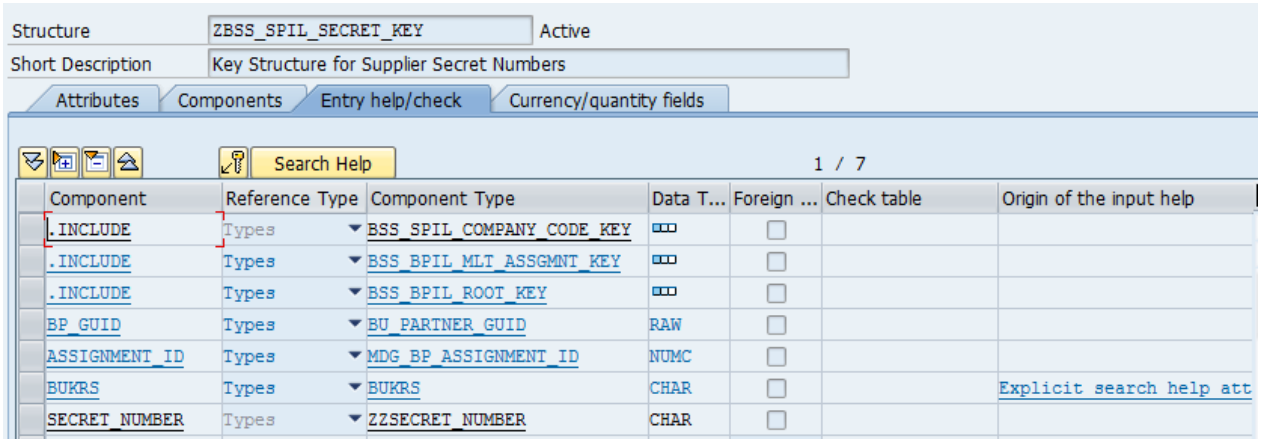
5.3.1 Structures

ZBSS_SPIL_SECRET



ZBSS_SPIL_SECRET_KEY





5.4 Source Code for genIL Model Class

IS_CHILD_CREATE_ALLOWED

```
method IF_GENIL_APPL_DYN_META_INFO~IS_CHILD_CREATE_ALLOWED.
*** This method controls if the creation of a supplier dependend object
*** is allowed or not.

* delegate to parent first
  super->if_genil_appl_dyn_meta_info~is_child_create_allowed( EXPORTING
                    iv_relation_name          = iv_relation_name
                    iv_relation_filter        = iv_relation_filter
                    iv_child_name             = iv_child_name
                  CHANGING
                    cv_parent_and_result = cv_parent_and_result ).

  CHECK me->mo_mdg_api IS BOUND
    AND cv_parent_and_result-success EQ abap_true.

  CASE iv_child_name.
    WHEN 'ZBS_SP_SECRET_NUMBERS'.
      cv_parent_and_result-success = abap_true. "It's always allowed

    WHEN
    OTHERS.

endmethod.
```

TRANSFORM_TO_ENTITY_KEY

```
method TRANSFORM_TO_ENTITY_KEY.
*** This method transforms a GenIL object ID into its entity key.
*** It requires the correct key structure in the chancing parameter.
  DATA:
    lr_genil_key TYPE REF TO data,
    lv_structure TYPE string.

  FIELD-SYMBOLS:
    <ls_genil_key> TYPE any,
```

```
    <lv_bp_id>      TYPE any,
    <lv_bp_guid>    TYPE any.

* limit handling to supplier objects
  CASE iv_object_name.
    WHEN 'ZBS_SP_SECRET_NUMBERS'.
*     get GenIL key
      TRY.
          lv_structure = me->object_model-
>get_key_struct_name( iv_object_name = iv_object_name ).
        CATCH cx_crm_unsupported_object.
          RETURN
      .ENDTRY.
      CREATE DATA lr_genil_key TYPE (lv_structure).
      ASSIGN lr_genil_key->* TO <ls_genil_key>.
      me->transform_to_object_key(
            EXPORTING
              iv_object_name = iv_object_name
              iv_object_id   = iv_object_id
            IMPORTING
              es_object_key  = <ls_genil_key> ).
*     map to entity key
      me->mo_typecasting->map(
            EXPORTING
              is_source_structure  = <ls_genil_key>
            CHANGING
              cs_target_structure  = cs_entity_key ).
*     BP ID requires special logic
      ASSIGN COMPONENT 'BP_HEADER' OF STRUCTURE cs_entity_key TO <lv_bp_id>.
      ASSIGN COMPONENT 'BP_GUID' OF STRUCTURE <ls_genil_key> TO <lv_bp_guid>.
      CHECK <lv_bp_id> IS ASSIGNED AND <lv_bp_guid> IS ASSIGNED.
      <lv_bp_id> = me->get_bp_id( iv_bp_guid = <lv_bp_guid> ).

    WHEN OTHERS.
*     call parent
      super->transform_to_entity_key(
              EXPORTING
                iv_object_name = iv_object_name
                iv_object_id   = iv_object_id
```

```
                CHANGING
                  cs_entity_key  = cs_entity_key ).
  ENDCASE.
endmethod.
```

6 Additional Information

6.1 Further Reading

6.1.1 Information on SAP MDG on SAP S/4HANA

- Exchange knowledge: SAP Community | Q&A | Blog
- Try SAP Master Data Governance on S/4HANA for free: Trial Version
- Try SAP Master Data Governance on S/4HANA on the SAP Cloud Appliance Library: S/4HANA 2022 FPS1
- Learn more: Latest Release | Help Portal | How-to Information | Key Presentations

6.1.2 SAP Roadmap Explorer

- Please see the roadmap for SAP Master Data Governance

6.1.3 Related Information

- Learn more: Floorplan Manager for Web Dynpro ABAP | How to Adapt FPM | FPM Blog | How-to Information | Service Mapping Tool | SAP S/4HANA Cookbook CVI

6.2 SAP Notes

In addition to the detailed explanations written in this document, please see the following SAP Notes for further important information.

Note	Description
2221398	MDG-BP/C/S/CA: (Un-)Supported Fields in Data Model BP
2847807	MDG-BP/C/S/CA: Usage of MDG Tools and Processes
2313368	Functional restrictions in MDG for Business Partner / Customer / Supplier with SAP Master Data Governance 9.0
2472845	Functional restrictions in MDG for Business Partner / Customer / Supplier with SAP Master Data Governance 9.1
2656712	Functional restrictions in MDG for Business Partner / Customer / Supplier in SAP Master Data Governance 9.2 and on SAP S/4HANA 1809
2816557	Functional restrictions in MDG for Business Partner / Customer / Supplier on SAP S/4HANA 1909
2925030	Functional restrictions in MDG for Business Partner / Customer / Supplier on SAP S/4HANA 2020
3070003	Functional restrictions in MDG for Business Partner / Customer / Supplier on SAP S/4HANA 2021
3220117	Functional restrictions in MDG for Business Partner / Customer / Supplier on SAP S/4HANA 2022
3374711	Functional restrictions in MDG for Business Partner / Customer / Supplier on SAP S/4HANA 2023
3043582	MDG Customer Connection 2020

3194967	MDG Customer Connection 2021 for S/4HANA 2022
3311039	MDG Customer Connection 2023
2479869	Usage of Lean Classification with SAP Master Data Governance
1619534	How to Create, Enhance and Adapt FPM Applications
1637249	MDG: Information for efficient message processing
2105467	MDG Performance
2561461	Scope of support for SAP Master Data Governance (MDG)
1637249	MDG: Information for efficient message processing