PUBLIC

How-To: Node (Table) Extension for Business Partner in MDG Consolidation and Mass Processing

Applicable Releases:

SAP MDG, Consolidation and SAP MDG, Mass Processing running on release SAP MDG 8.0 or higher and SAP S/4HANA 1610 or higher.

Version 1.2

January 2023

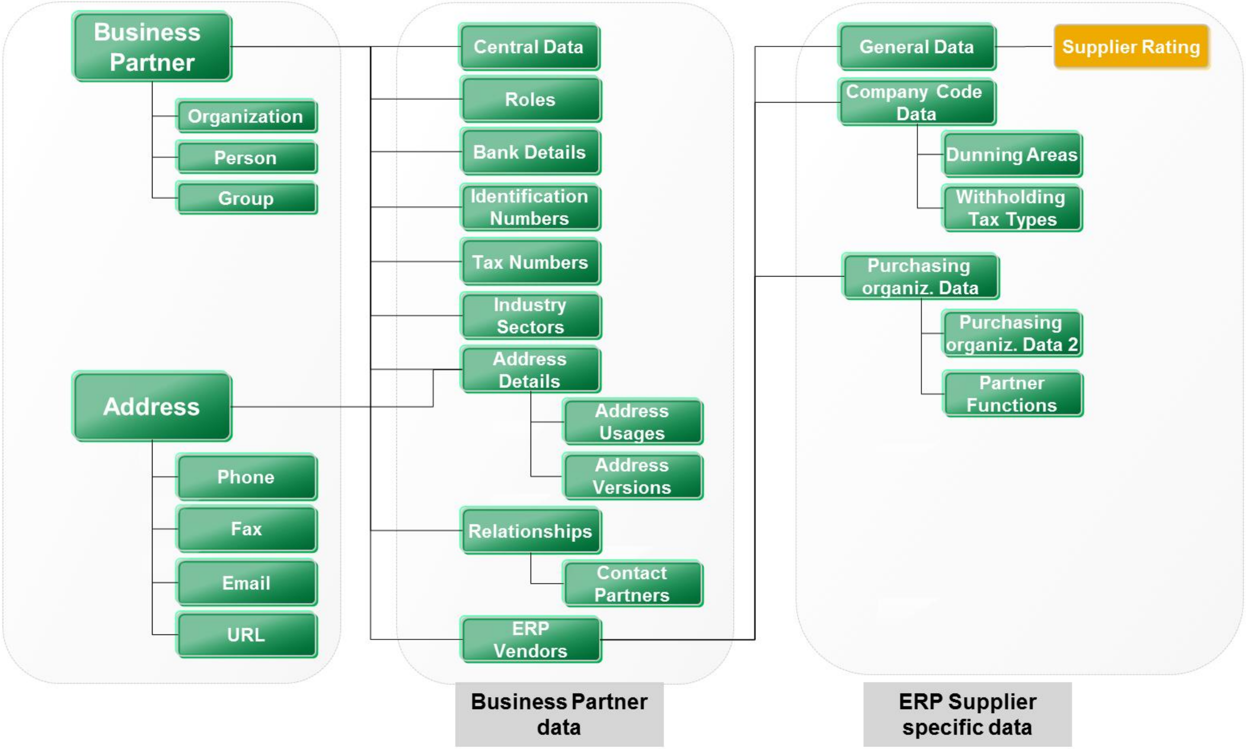
Document History

Document Version	Description
1.0	First official release of this guide (Month 2023)
1.1	Minor changes
1.2	Formatting changed

THE BEST RUN SAP®

1. BUSINESS SCENARIO

To process supplier data within Business Partner consolidation or mass processing, a customer-specific table Supplier Ratings is required. During a consolidation or mass process, the additional supplier rating data will be loaded into source tables and/or process tables and be processed along with standard supplier data. After activation, those attributes will be stored in a custom Z-table.



2. BACKGROUND INFORMATION

Before a new table can be handled properly in MDG, Consolidation or MDG, Mass Processing, the following extensions must be made:

- Redefine and extend MDG, Consolidation classes
- Extend MDG process model
- Extend MDG for customer tables to activate the data into an MDG change request

3. STEP BY STEP EXPLANATION

The following explanation shows you how to add a new node/table to the Business Partner data model for MDG Consolidation or MDG Mass Processing. The following steps provide details on how to extend the data model by adding a new table ZSUPPLIER_RATING for ERP Supplier.
For details of the individual DDIC-objects, refer to the Appendix section *Data Dictionary Objects*.

3.1. Extend ERP supplier data model in MDG, central governance

If the consolidated supplier rating data will be passed along with Business Partner and ERP Supplier data to a Change Request within MDG, MDG must be extended accordingly. Those extensions are prerequisite and are not part of this guide.

3.2. Redefine Business Partner model classes

To use and access data in the newly created source and process tables within a process, the following classes must be created. They must inherit from the given superclasses and certain methods must be redefined.
A detailed source code example is provided in the Appendix.
Also note that, as already stated, if the data will be passed to a Change Request, the corresponding MDG extensions must have been implemented. They are not part of this guide.

3.2.1. Create and redefine Business Partner data access class

Create a new class ZCL_MDC_DATA_BP that inherits from class CL_MDC_DATA_BP. The new class will then be used in consolidation processes involving Process Model 147 (Business Partner). The following methods must be redefined:

- TABLE_NAME_BY_TYPE
  A redefinition of this method is only required if the new table for consolidation has 16 characters, like ZSUPPLIER_RATING, or if own names for source and process tables will be used.
- APPEND_ACTIVE_RECORDS

Fields using a “Large Object Binary” data type
If your custom field uses a “Large Object Binary” related data type (e.g. a string, blob, raw binary or similar), you need to redefine one more method in your custom Business Partner Data Access class, namely:

- IF_MDC_DATA~ CONTAINS_LOB_DATA
  A redefinition of this method is only required if the new custom field is using a “large binary object” data type. In this case, ensure that the method returns “abap_true” for the affected table(s).

3.2.2. Create and redefine Business Partner model implementation class

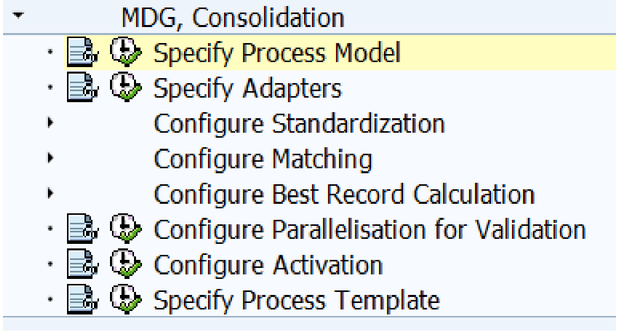
Create a new class ZCL_MDC_MODEL_BP that inherits from class CL_MDC_MODEL_BP. The new class will then be used in processes involving Process Model 147 (Business Partner). The following methods must be redefined:

- READ_ALL_DATA
- MAP_EXTENSIONS_2API
- SAVE_BUSINESS_PARTNER_ACTIVE

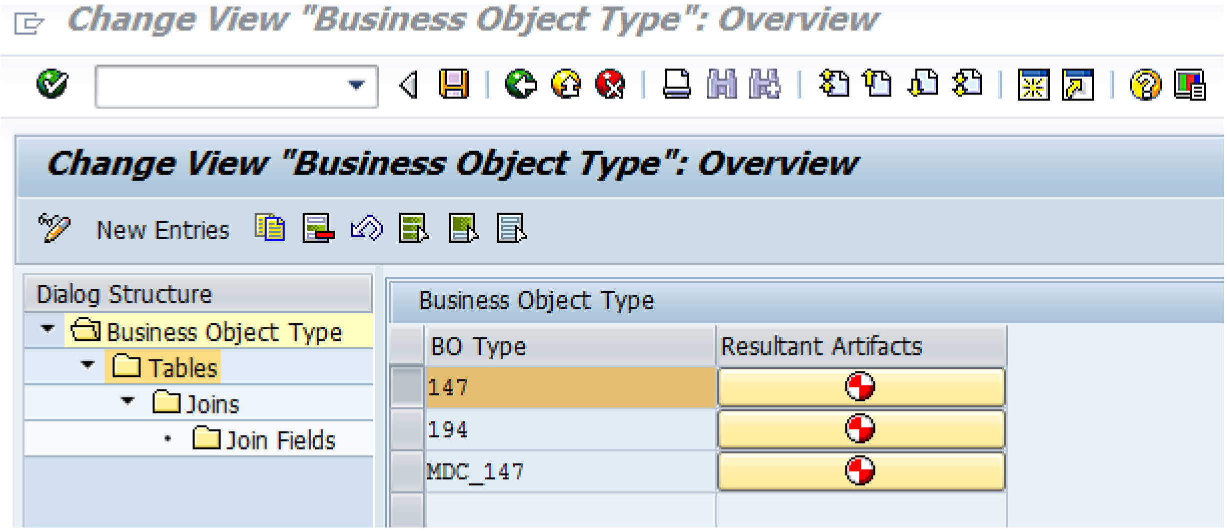
3.3. Extend process model

To extend the Process Model with a new table, the contents of View Cluster VC_MDC_MODEL must be changed. This view cluster contains the process model, which includes all relevant tables of an object and their relations.

1. Start IMG Activity *Specify Process Model* in the implementation guide for Consolidation and Mass Processing (Transaction MDCIMG).

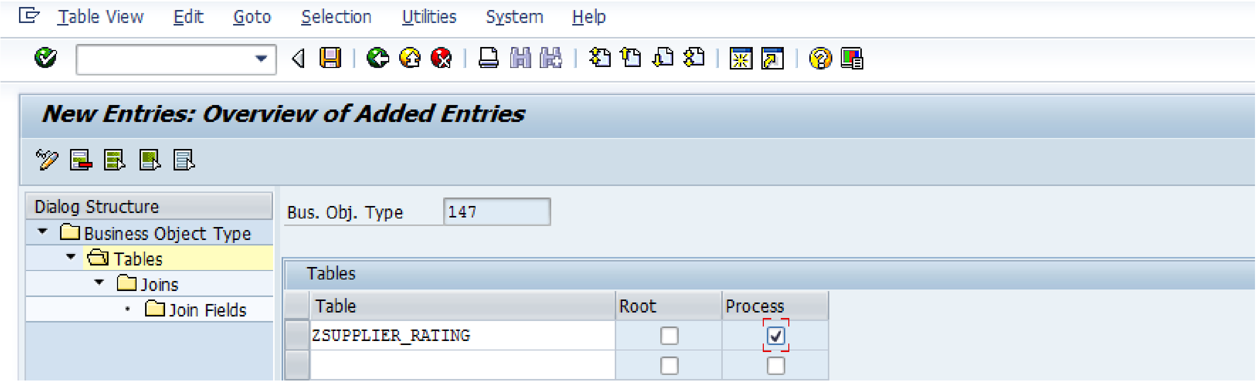


2. Select Business Object Type 147 (Business Partner) and navigate to the sub-node Tables.



BO Type	Resultant Artifacts
147	◉
194	◉
MDC_147	◉

Add a new process table ZSUPPLIER_RATING to the process model. Add the name of the database table to the *Table* column and mark the *Process* column.



Bus. Obj. Type 147

Table	Root	Process
ZSUPPLIER_RATING	☐	☑
	☐	☐

3. Save the changes.

4. Add a new join to table LFA1 by selecting the table line and navigating to Joins. Add new entry ZSUPPLIER_RATING and mark *Process* and *Active*.

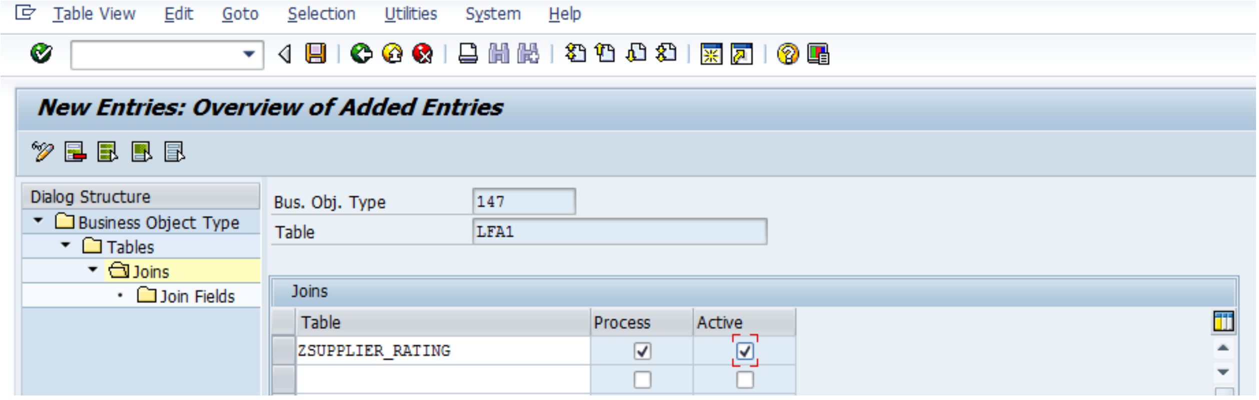


Table View Edit Goto Selection Utilities System Help
New Entries: Overview of Added Entries
Dialog Structure
Business Object Type
Tables
Joins
Join Fields

Bus. Obj. Type: 147

Table: LFA1

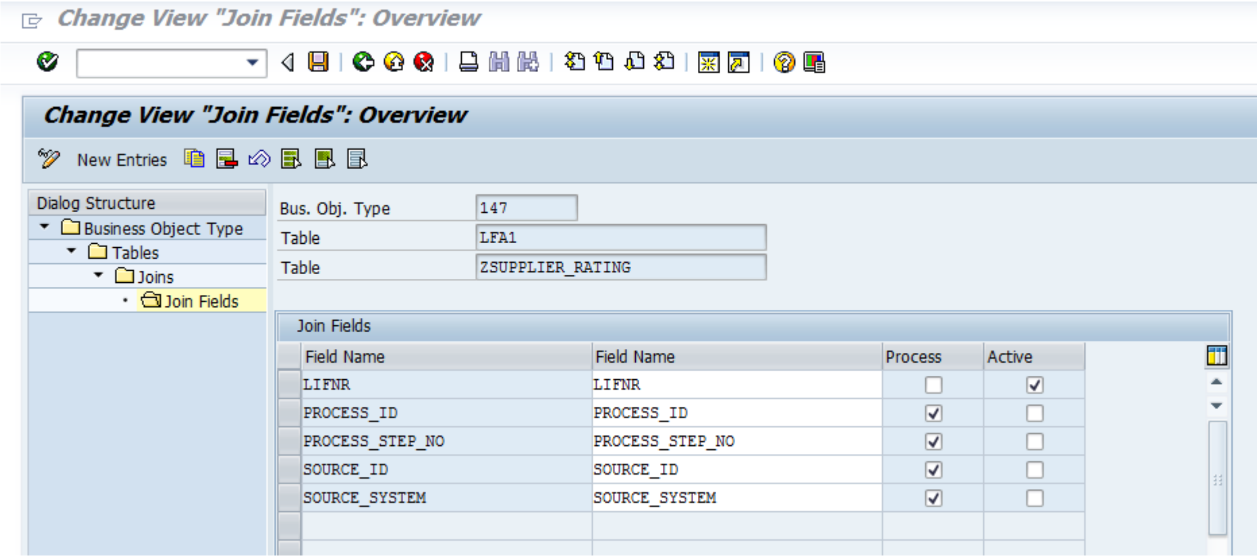
Joins

Table	Process	Active
ZSUPPLIER_RATING	☑	☑
	□	□

5. Save the changes.

6. Select the added Join ZSUPPLIER_RATING and navigate to *Join Fields* in order to link the parent table LFA1 and child table ZSUPPLIER_RATING. Add the following records into *Join Fields*:

Field Name	Field Name	Process	Active
LIFNR	LIFNR		X
PROCESS_ID	PROCESS_ID	X	
PROCESS_STEP_NO	PROCESS_STEP_NO	X	
SOURCE_ID	SOURCE_ID	X	
SOURCE_SYSTEM	SOURCE_SYSTEM	X	


Change View "Join Fields": Overview
Change View "Join Fields": Overview
New Entries
Dialog Structure
Business Object Type
Tables
Joins
Join Fields

Bus. Obj. Type: 147

Table: LFA1

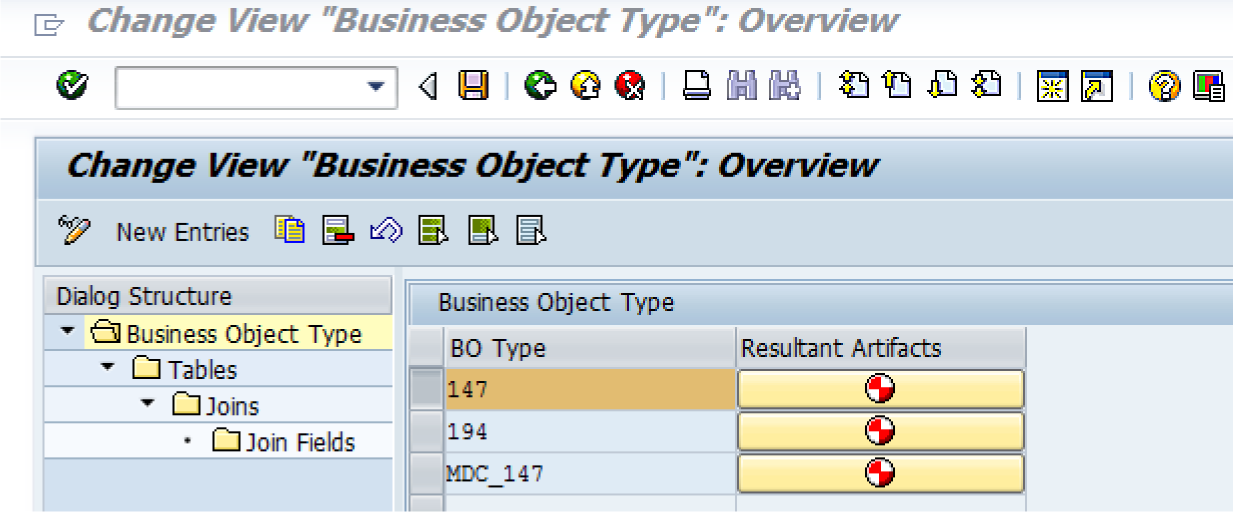
Table: ZSUPPLIER_RATING

Join Fields

Field Name	Field Name	Process	Active
LIFNR	LIFNR	□	☑
PROCESS_ID	PROCESS_ID	☑	□
PROCESS_STEP_NO	PROCESS_STEP_NO	☑	□
SOURCE_ID	SOURCE_ID	☑	□
SOURCE_SYSTEM	SOURCE_SYSTEM	☑	□

7. Save the changes.

8. Generate artifacts for new table ZSUPPLIER_RATING for Business Object Type 147.

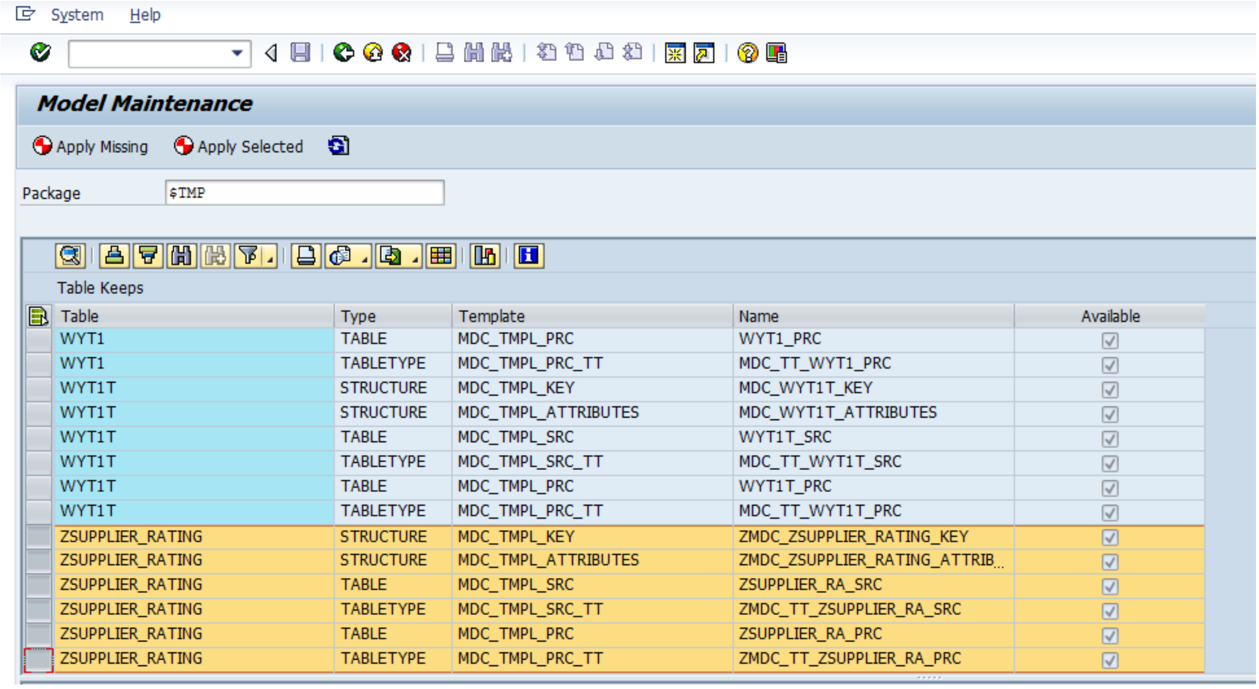


Change View "Business Object Type": Overview

Business Object Type

BO Type	Resultant Artifacts
147	●
194	●
MDC_147	●

9. Select all rows containing the new table name ZSUPPLIER_RATING, choose a package in which the new objects shall be created, and choose *Apply Selected*.



Package: $TMP

Table Keeps

Table	Type	Template	Name	Available
WYT1	TABLE	MDC_TMPL_PRC	WYT1_PRC	☑
WYT1	TABLETYPE	MDC_TMPL_PRC_TT	MDC_TT_WYT1_PRC	☑
WYT1T	STRUCTURE	MDC_TMPL_KEY	MDC_WYT1T_KEY	☑
WYT1T	STRUCTURE	MDC_TMPL_ATTRIBUTES	MDC_WYT1T_ATTRIBUTES	☑
WYT1T	TABLE	MDC_TMPL_SRC	WYT1T_SRC	☑
WYT1T	TABLETYPE	MDC_TMPL_SRC_TT	MDC_TT_WYT1T_SRC	☑
WYT1T	TABLE	MDC_TMPL_PRC	WYT1T_PRC	☑
WYT1T	TABLETYPE	MDC_TMPL_PRC_TT	MDC_TT_WYT1T_PRC	☑
ZSUPPLIER_RATING	STRUCTURE	MDC_TMPL_KEY	ZMDC_ZSUPPLIER_RATING_KEY	☑
ZSUPPLIER_RATING	STRUCTURE	MDC_TMPL_ATTRIBUTES	ZMDC_ZSUPPLIER_RATING_ATTRIB...	☑
ZSUPPLIER_RATING	TABLE	MDC_TMPL_SRC	ZSUPPLIER_RA_SRC	☑
ZSUPPLIER_RATING	TABLETYPE	MDC_TMPL_SRC_TT	ZMDC_TT_ZSUPPLIER_RA_SRC	☑
ZSUPPLIER_RATING	TABLE	MDC_TMPL_PRC	ZSUPPLIER_RA_PRC	☑
ZSUPPLIER_RATING	TABLETYPE	MDC_TMPL_PRC_TT	ZMDC_TT_ZSUPPLIER_RA_PRC	☑

Now all relevant DDIC-objects relevant have been created, including:

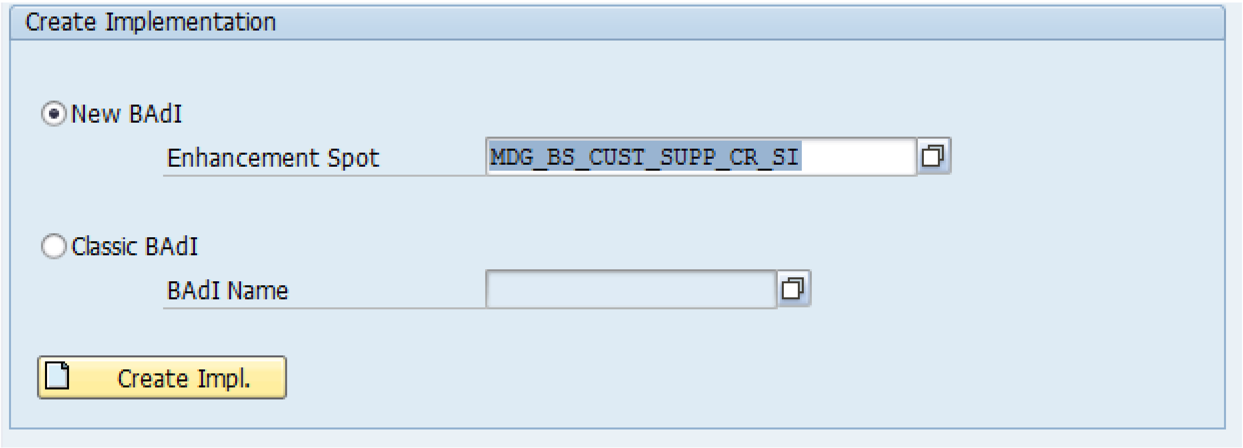
- Source database table (ZSUPPLIER_RA_SRC) & corresponding table type
- Process database table (ZSUPPLIER_RA_PRC) & corresponding table type
- Key and attribute structures

3.4. Extend MDG for writing data into MDG change request

The prerequisite extensibility guide for MDG “Extend MDG-S / MDG-C Data Model by a New Field (Reuse Option)” describes most of the steps needed for extending MDG-S / MDG-C with a new table. This section here focuses on the back-end part of such an extension that is required in order to bring activated data of a new table into a change request.
A detailed source code example is provided in the Appendix.

1. Create a new BAdI Implementation for BAdI Definition MDG_BS_CUST_SUPP_CR_SI (Enhancement Spot MDG_BS_CUST_SUPP_CR_SI) using transaction SE19.

a. Select New BAdI, Enhancement Sport MDG_BS_CUST_SUPP_CR_SI and choose *Create Impl*.



Create Implementation

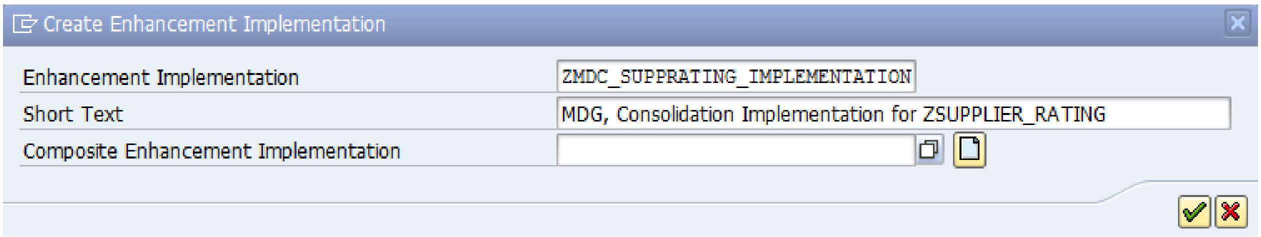
● New BAdI

Enhancement Spot	MDG_BS_CUST_SUPP_CR_SI

○ Classic BAdI

BAdI Name	

b. Create new Enhancement Implementation ZMDC_SUPPRATING_IMPLEMENTATION and enter a short text.

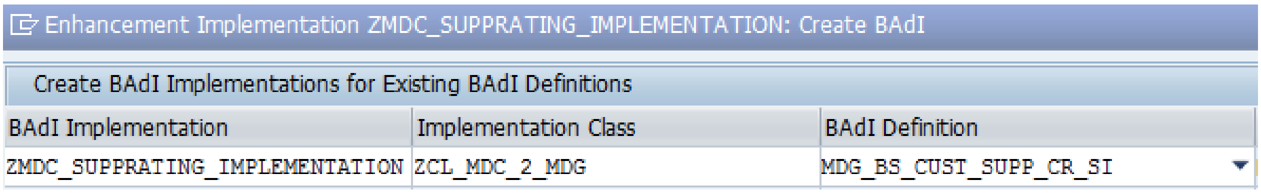


Create Enhancement Implementation

Enhancement Implementation	ZMDC_SUPPRATING_IMPLEMENTATION
Short Text	MDG, Consolidation Implementation for ZSUPPLIER_RATING
Composite Enhancement Implementation	

c. Create BAdI Implementation.

BAdI Implementation	ZMDC_SUPPRATING_IMPLEMENTATION
Implementation Class	ZCL_MDC_2_MDG
BAdI Definition	MDG_BS_CUST_SUPP_CR_SI



Enhancement Implementation ZMDC_SUPPRATING_IMPLEMENTATION: Create BAdI

Create BAdI Implementations for Existing BAdI Definitions

BAdI Implementation	Implementation Class	BAdI Definition
ZMDC_SUPPRATING_IMPLEMENTATION	ZCL_MDC_2_MDG	MDG_BS_CUST_SUPP_CR_SI

d. Activate Enhancement Implementation.

2. Implement the following methods of BAdI Implementation class ZCL_MDC_2_MDG:

   - CONSTRUCTOR
   - CALCULATE_DELETIONS
   - IF_MDG_BS_CUST_SUPP_CR_SI~SET_SUPPLIER_DATA_2_STA

   Note that at this point the MDG data model must have already been extended with a new entity for ZSUPPLIER_RATING. In this example, a new entity ZSUPP_RAC has been created for that case.

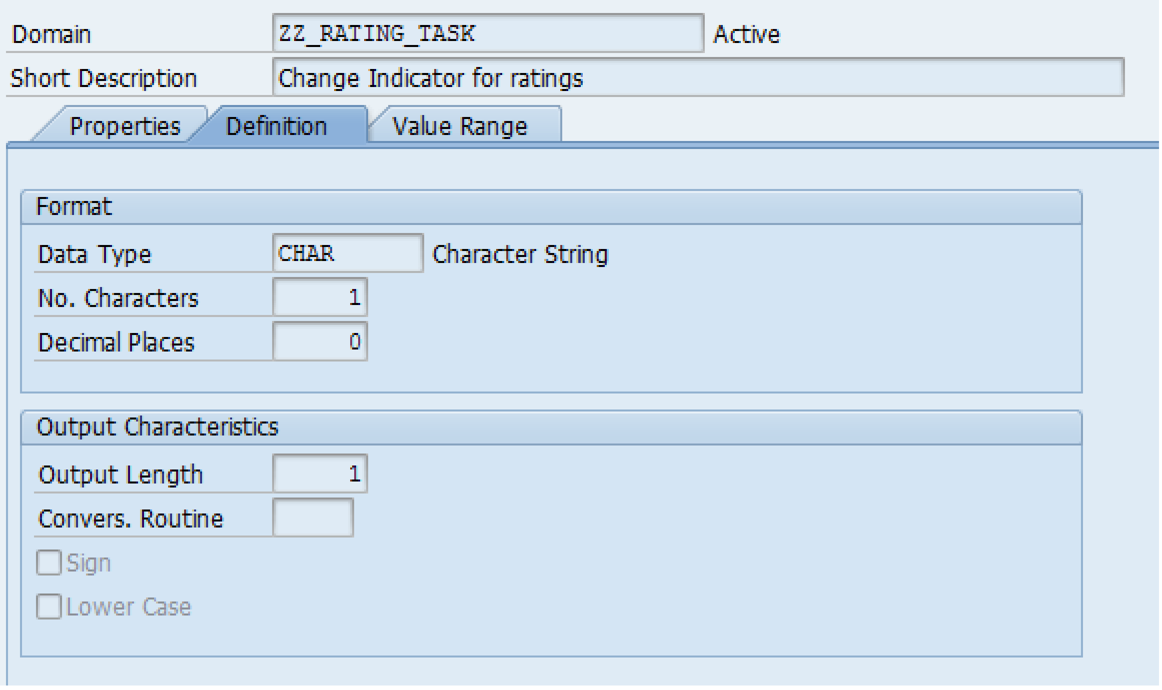
4. APPENDIX

4.1. Data Dictionary Objects

This section contains all DDIC-objects that have not been automatically created / generated and are used in the example implementation.

4.1.1. Domains

ZZ_RATING_TASK



Domain: ZZ_RATING_TASK Active

Short Description: Change Indicator for ratings

Format

Data Type: CHAR Character String

No. Characters: 1

Decimal Places: 0

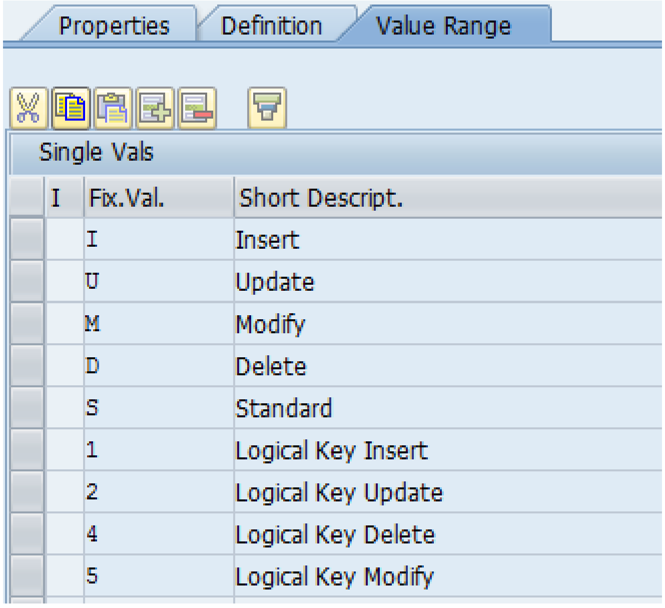
Output Characteristics

Output Length: 1

Convers. Routine:

☐ Sign

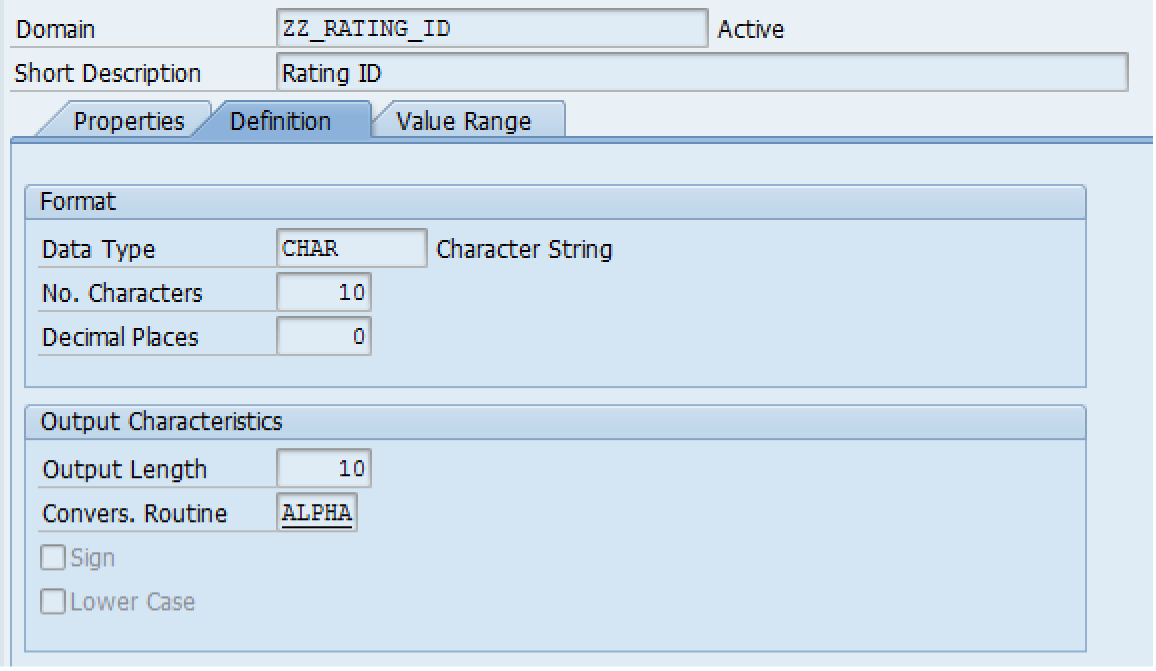
☐ Lower Case



Single Vals

I	Fix.Val.	Short Descript.
	I	Insert
	U	Update
	M	Modify
	D	Delete
	S	Standard
	1	Logical Key Insert
	2	Logical Key Update
	4	Logical Key Delete
	5	Logical Key Modify

ZZ_RATING_ID



Domain	ZZ_RATING_ID	Active
Short Description	Rating ID	

Format

Data Type	CHAR	Character String
No. Characters	10	
Decimal Places	0	

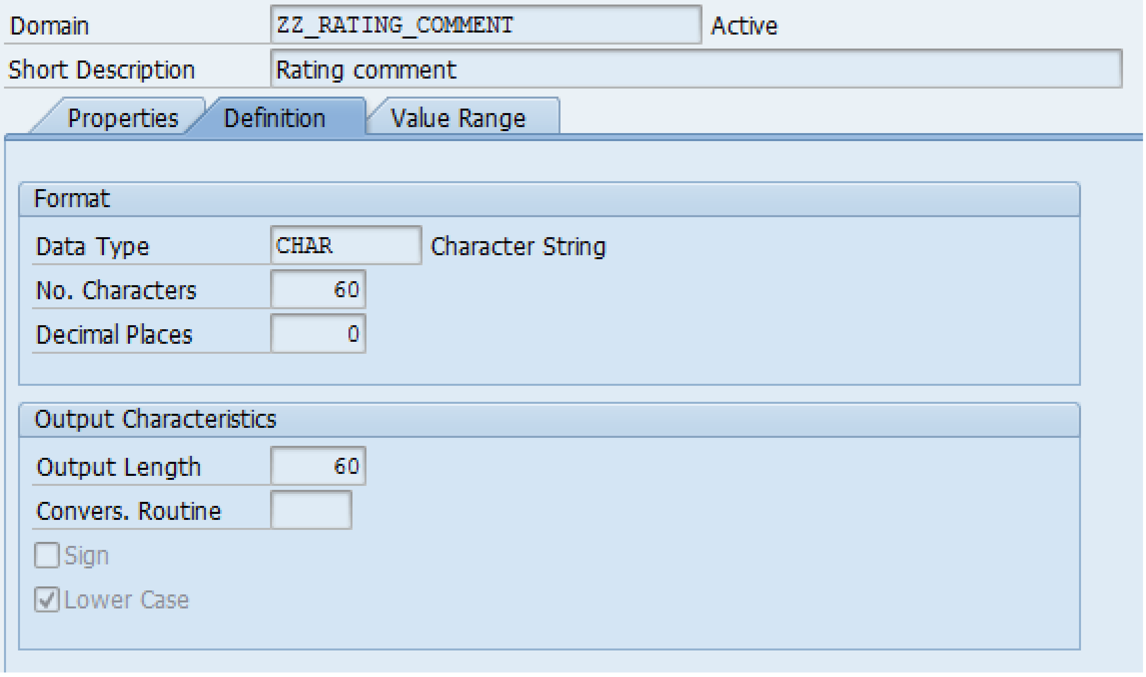
Output Characteristics

Output Length	10
Convers. Routine	ALPHA

□ Sign

□ Lower Case

ZZ_RATING_COMMENT



Domain	ZZ_RATING_COMMENT	Active
Short Description	Rating comment	

Format

Data Type	CHAR	Character String
No. Characters	60	
Decimal Places	0	

Output Characteristics

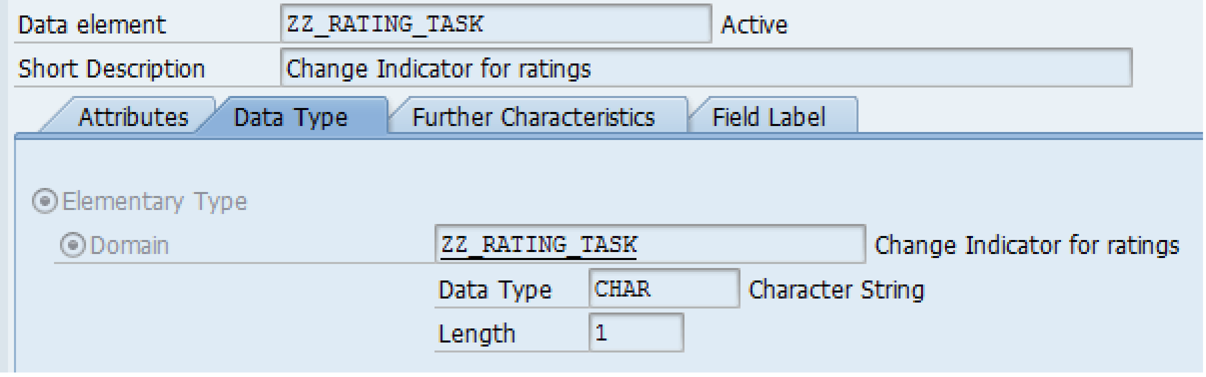
Output Length	60
Convers. Routine	

□ Sign

☑ Lower Case

4.1.2. Data Elements

ZZ_RATING_TASK



Data element	ZZ_RATING_TASK	Active
Short Description	Change Indicator for ratings	

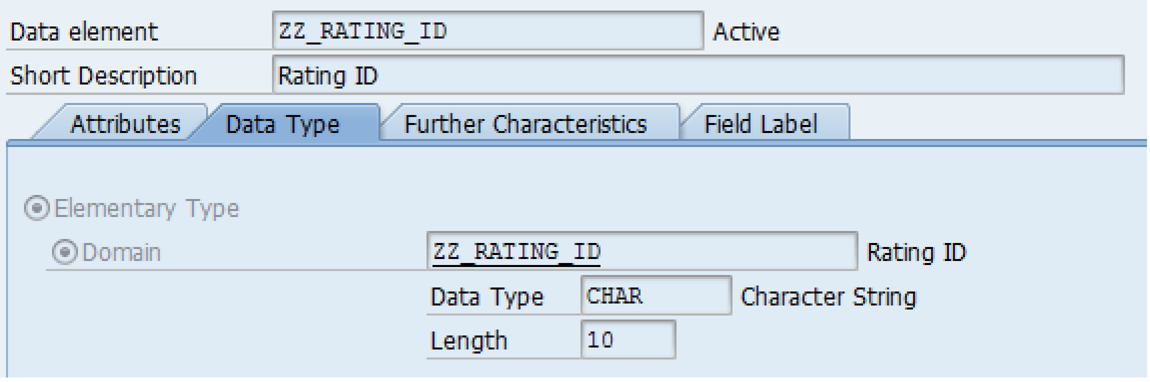
● Elementary Type

● Domain ZZ_RATING_TASK Change Indicator for ratings

Data Type CHAR Character String

Length 1

ZZ_RATING_ID



Data element	ZZ_RATING_ID	Active
Short Description	Rating ID	

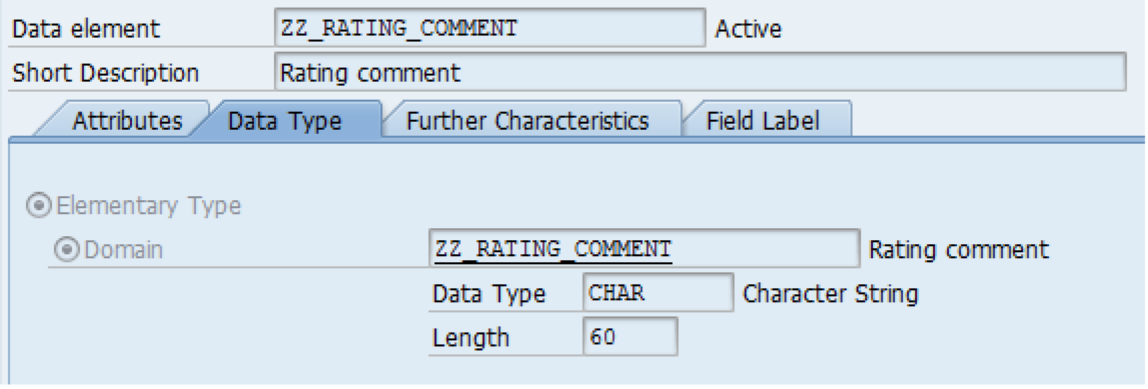
● Elementary Type

● Domain ZZ_RATING_ID Rating ID

Data Type CHAR Character String

Length 10

ZZ_RATING_COMMENT



Data element	ZZ_RATING_COMMENT	Active
Short Description	Rating comment	

● Elementary Type

● Domain ZZ_RATING_COMMENT Rating comment

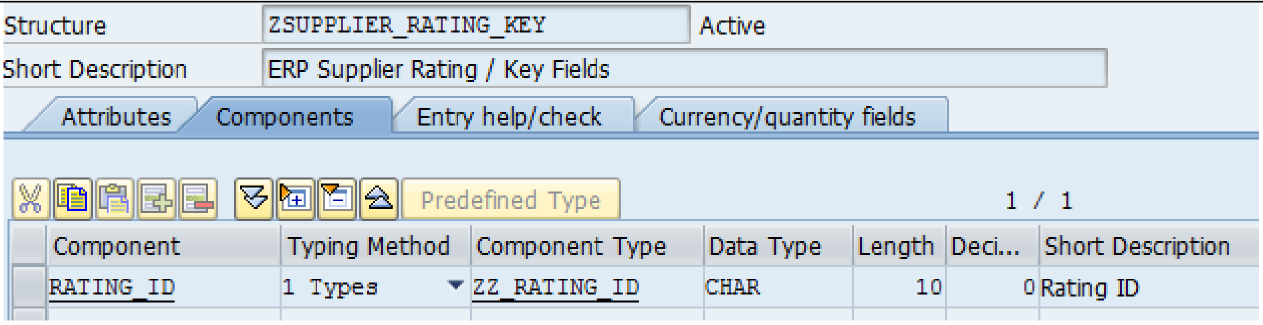
Data Type CHAR Character String

Length 60

4.1.3. Structures

The structures listed here are relevant for updating the database table ZSUPPLIER_RATING. The assumption is that those structures already exist in a customer system.

ZSUPPLIER_RATING_KEY

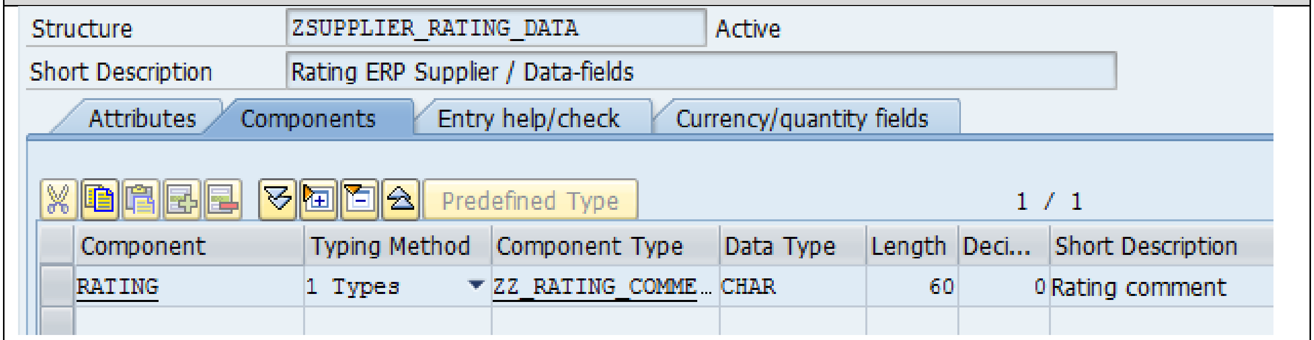


Structure: ZSUPPLIER_RATING_KEY Active

Short Description: ERP Supplier Rating / Key Fields

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
RATING_ID	1 Types	ZZ_RATING_ID	CHAR	10	0	Rating ID

ZSUPPLIER_RATING_DATA

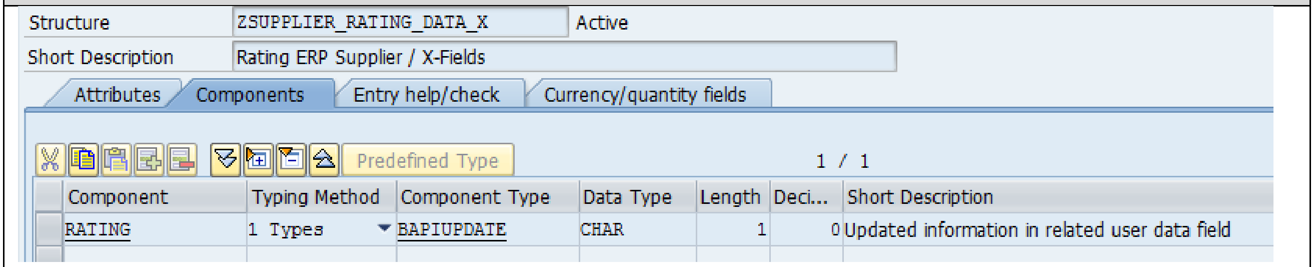


Structure: ZSUPPLIER_RATING_DATA Active

Short Description: Rating ERP Supplier / Data-fields

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
RATING	1 Types	ZZ_RATING_COMME...	CHAR	60	0	Rating comment

ZSUPPLIER_RATING_DATA_X

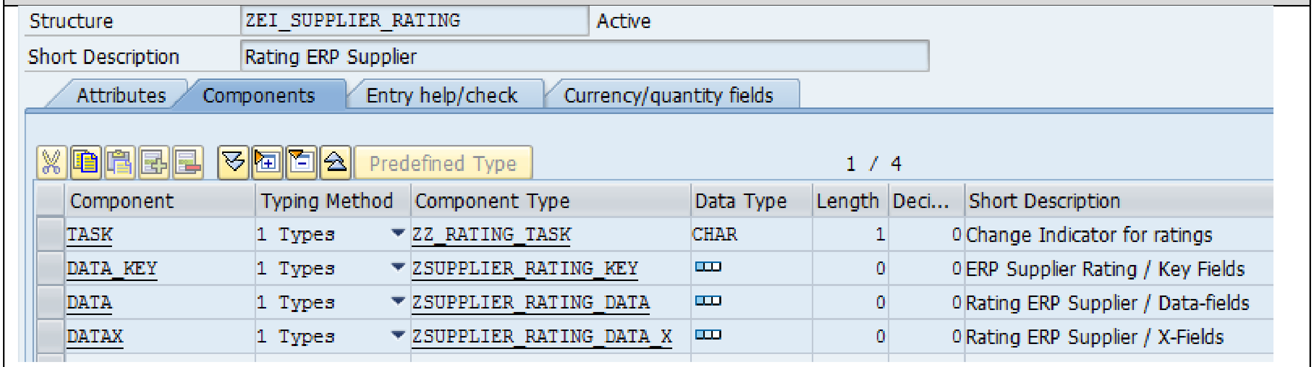


Structure: ZSUPPLIER_RATING_DATA_X Active

Short Description: Rating ERP Supplier / X-Fields

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
RATING	1 Types	BAPIUPDATE	CHAR	1	0	Updated information in related user data field

ZEI_SUPPLIER_RATING

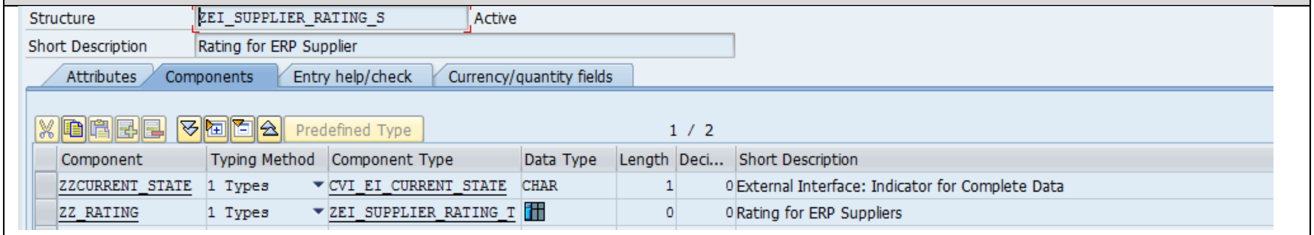


Structure: ZEI_SUPPLIER_RATING Active

Short Description: Rating ERP Supplier

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
TASK	1 Types	ZZ_RATING_TASK	CHAR	1	0	Change Indicator for ratings
DATA_KEY	1 Types	ZSUPPLIER_RATING_KEY	▭	0	0	ERP Supplier Rating / Key Fields
DATA	1 Types	ZSUPPLIER_RATING_DATA	▭	0	0	Rating ERP Supplier / Data-fields
DATAX	1 Types	ZSUPPLIER_RATING_DATA_X	▭	0	0	Rating ERP Supplier / X-Fields

ZEI_SUPPLIER_RATING_S

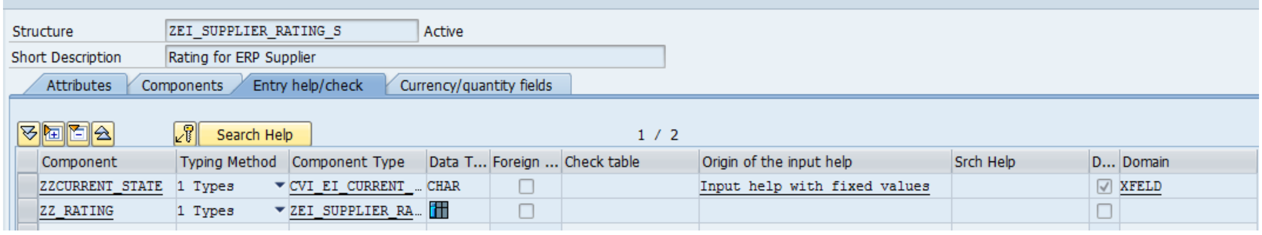


Structure	ZEI_SUPPLIER_RATING_S	Active
Short Description	Rating for ERP Supplier	

Attributes / Components / Entry help/check / Currency/quantity fields

Predefined Type 1 / 2

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
ZZCURRENT_STATE	1 Types	CVI_EI_CURRENT_STATE	CHAR	1	0	External Interface: Indicator for Complete Data
ZZ_RATING	1 Types	ZEI_SUPPLIER_RATING_T	▦	0	0	Rating for ERP Suppliers



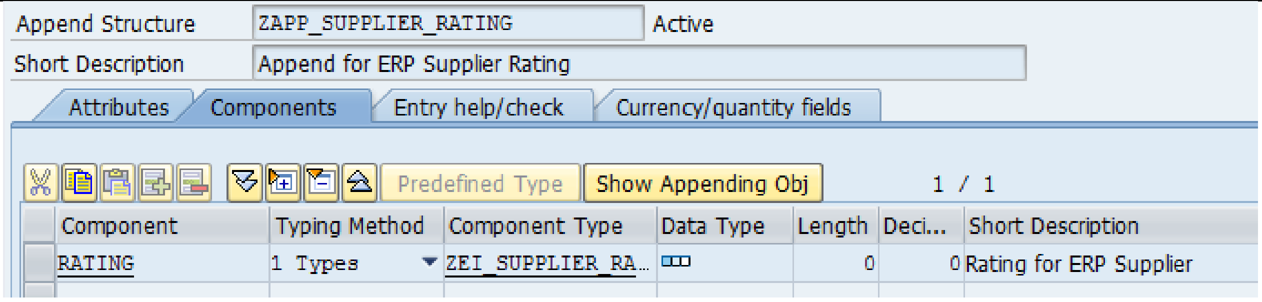
Structure	ZEI_SUPPLIER_RATING_S	Active
Short Description	Rating for ERP Supplier	

Attributes / Components / Entry help/check / Currency/quantity fields

Search Help 1 / 2

Component	Typing Method	Component Type	Data T...	Foreign ...	Check table	Origin of the input help	Srch Help	D...	Domain
ZZCURRENT_STATE	1 Types	CVI_EI_CURRENT_...	CHAR	□		Input help with fixed values		☑	XFELD
ZZ_RATING	1 Types	ZEI_SUPPLIER_RA...	▦	□				□	

ZAPP_SUPPLIER_RATING

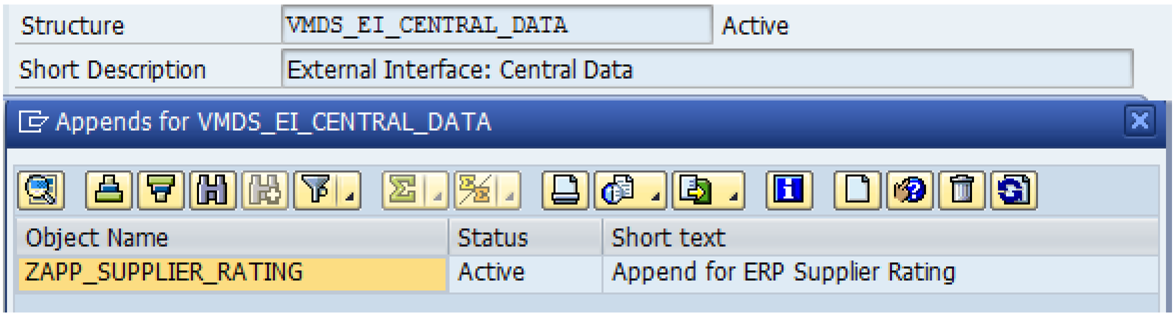


Append Structure	ZAPP_SUPPLIER_RATING	Active
Short Description	Append for ERP Supplier Rating	

Attributes / Components / Entry help/check / Currency/quantity fields

Predefined Type / Show Appending Obj 1 / 1

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
RATING	1 Types	ZEI_SUPPLIER_RA...	▭	0	0	Rating for ERP Supplier



Structure	VMDS_EI_CENTRAL_DATA	Active
Short Description	External Interface: Central Data	

Appends for VMDS_EI_CENTRAL_DATA

Object Name	Status	Short text
ZAPP_SUPPLIER_RATING	Active	Append for ERP Supplier Rating

4.1.4. *Table types*

ZEI_SUPPLIER_RATING_T

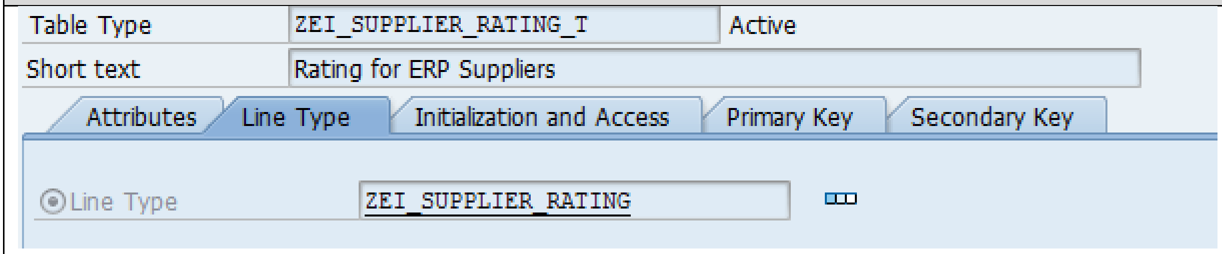


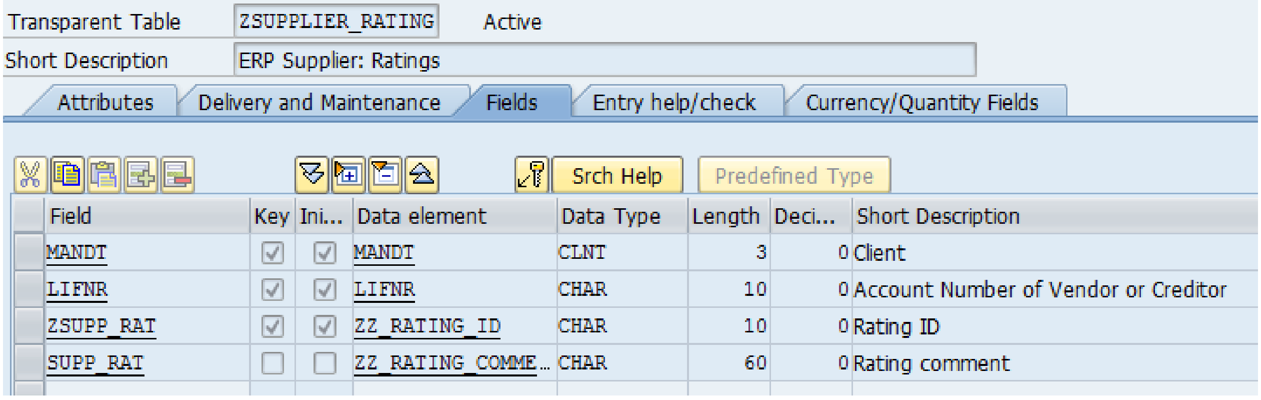
Table Type	ZEI_SUPPLIER_RATING_T	Active
Short text	Rating for ERP Suppliers	

Attributes / Line Type / Initialization and Access / Primary Key / Secondary Key

● Line Type ZEI_SUPPLIER_RATING

4.1.5. *Database tables*

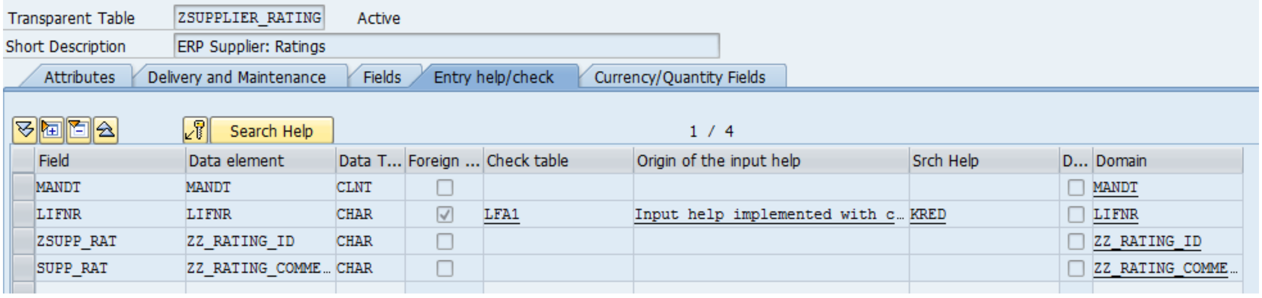
ZSUPPLIER_RATING



Transparent Table ZSUPPLIER_RATING Active

Short Description ERP Supplier: Ratings

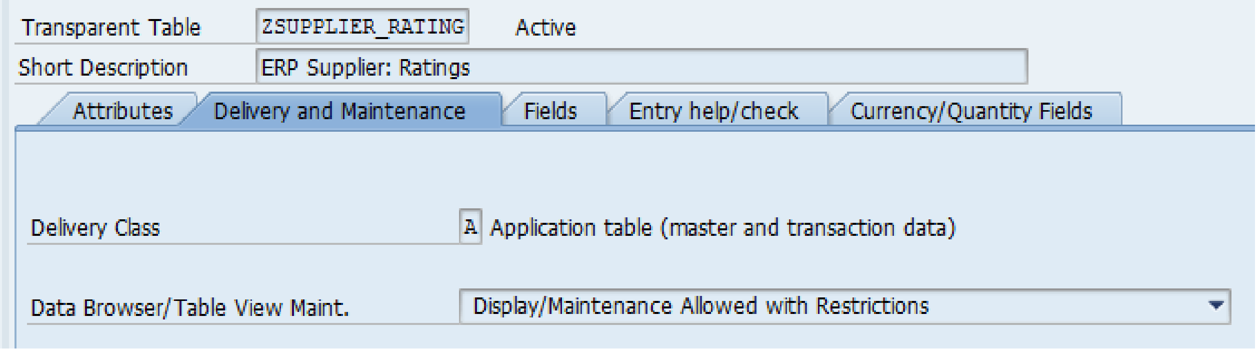
Field	Key	Ini...	Data element	Data Type	Length	Deci...	Short Description
MANDT	☑	☑	MANDT	CLNT	3	0	Client
LIFNR	☑	☑	LIFNR	CHAR	10	0	Account Number of Vendor or Creditor
ZSUPP_RAT	☑	☑	ZZ_RATING_ID	CHAR	10	0	Rating ID
SUPP_RAT	□	□	ZZ_RATING_COMME…	CHAR	60	0	Rating comment



Transparent Table ZSUPPLIER_RATING Active

Short Description ERP Supplier: Ratings

Field	Data element	Data T...	Foreign ...	Check table	Origin of the input help	Srch Help	D...	Domain
MANDT	MANDT	CLNT	□				□	MANDT
LIFNR	LIFNR	CHAR	☑	LFA1	Input help implemented with c…	KRED	□	LIFNR
ZSUPP_RAT	ZZ_RATING_ID	CHAR	□				□	ZZ_RATING_ID
SUPP_RAT	ZZ_RATING_COMME…	CHAR	□				□	ZZ_RATING_COMME…



Transparent Table ZSUPPLIER_RATING Active

Short Description ERP Supplier: Ratings

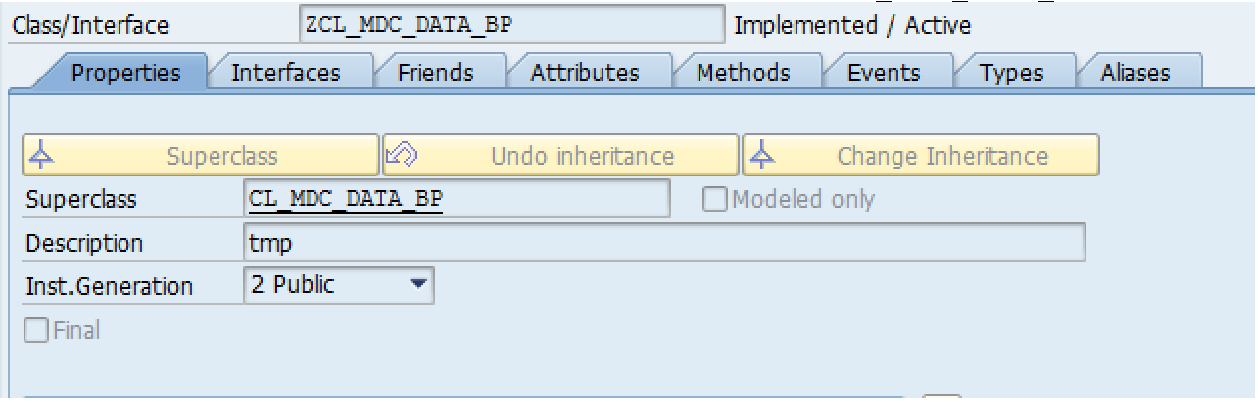
Delivery Class A Application table (master and transaction data)

Data Browser/Table View Maint. Display/Maintenance Allowed with Restrictions

4.2. Sample Source Code

4.2.1. *Sample source code data access class*

The customer data access class for Business Partner must inherit from CL_MDC_DATA_BP.



Class/Interface ZCL_MDC_DATA_BP Implemented / Active

Superclass CL_MDC_DATA_BP □ Modeled only

Description tmp

Inst.Generation 2 Public

□ Final

Method Redefinitions

TABLE_NAME_BY_TYPE

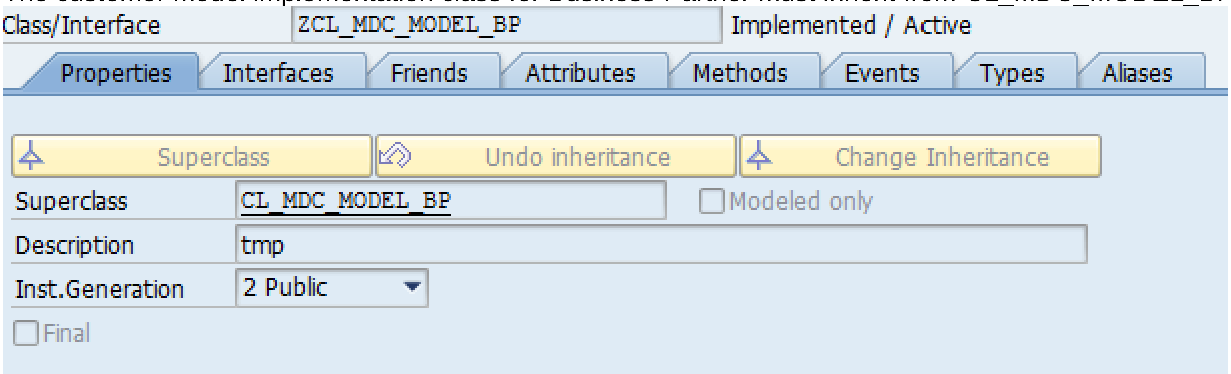
```
METHOD if_mdc_data~table_name_by_type.
  rv_table_name = super->if_mdc_data~table_name_by_type( iv_type ).
  CHECK me->table_name = 'ZSUPPLIER_RATING'.
  CASE iv_type.
    WHEN if_mdc_data=>gc_type-source.
      rv_table_name+12(4) = if_mdc_data=>gc_suffix-source.
    WHEN if_mdc_data=>gc_type-process.
      rv_table_name+12(4) = if_mdc_data=>gc_suffix-process.
  ENDCASE.
ENDMETHOD.
```

APPEND_ACTIVE_RECORDS

```
METHOD if_mdc_data~append_active_records.
  DATA lt_bp_source_keys TYPE tt_bp_source_keys.

  CHECK me->process_relevant = abap_true.
  CHECK it_source_keys IS NOT INITIAL.
  lt_bp_source_keys = CONV #( it_source_keys ).

  CASE me->table_name.
    WHEN 'ZSUPPLIER_RATING'.
      me->append_supplier_data( lt_bp_source_keys ).
    WHEN OTHERS.
      super->if_mdc_data~append_active_records( it_source_keys ).
  ENDCASE.

ENDMETHOD.
```

4.2.2. Sample source code model implementation class

The customer model implementation class for Business Partner must inherit from CL_MDC_MODEL_BP.



Class/Interface ZCL_MDC_MODEL_BP Implemented / Active

Superclass CL_MDC_MODEL_BP ☐ Modeled only

Description tmp

Inst.Generation 2 Public

☐ Final

Method Redefinitions

READ_ALL_DATA

```
METHOD read_all_data.

  CALL METHOD super->read_all_data
    EXPORTING
      it_source_keys    = it_source_keys
      iv_package_number = iv_package_number.
    me->mr_zsupplier_rating_prc = CAST #( me->object( 'ZSUPPLIER_RATING' )->
read( it_source_keys = it_source_keys iv_package_number = iv_package_number ) ).

  ENDMETHOD.
```

MAP_EXTENSIONS_2API

```
  METHOD map_extensions_2api.
    DATA ls_supp_rating_cvi TYPE zei_supplier_rating.

    FIELD-SYMBOLS <ls_supp_rating> TYPE zsupplier_ra_prc.

    " Extension Supplier Rating:
    DATA(lt_supp_rating)  = VALUE zmdc_tt_zsupplier_ra_prc( FOR zsupplier_rating IN me
->mr_zsupplier_rating_prc->*
      WHERE ( process_key = is_but000_prc-process_key ) ( zsupplier_rating ) ).

    LOOP AT lt_supp_rating ASSIGNING <ls_supp_rating>.
      IF cs_cvi_ei_extrn-vendor-header-object_task = 'I'.
        ls_supp_rating_cvi-task = 'I'.
      ELSE.
        ls_supp_rating_cvi-task = 'M'.
      ENDIF.

      ls_supp_rating_cvi-data_key-rating_id = <ls_supp_rating>-zsupp_rat.
      ls_supp_rating_cvi-data-rating = <ls_supp_rating>-supp_rat.
      ls_supp_rating_cvi-datax-rating = abap_true.

      APPEND ls_supp_rating_cvi TO cs_cvi_ei_extrn-vendor-central_data-rating-
zz_rating.
      CLEAR ls_supp_rating_cvi.
    ENDLOOP.

    cs_cvi_ei_extrn-vendor-central_data-rating-zzcurrent_state = abap_true.

  ENDMETHOD.
```

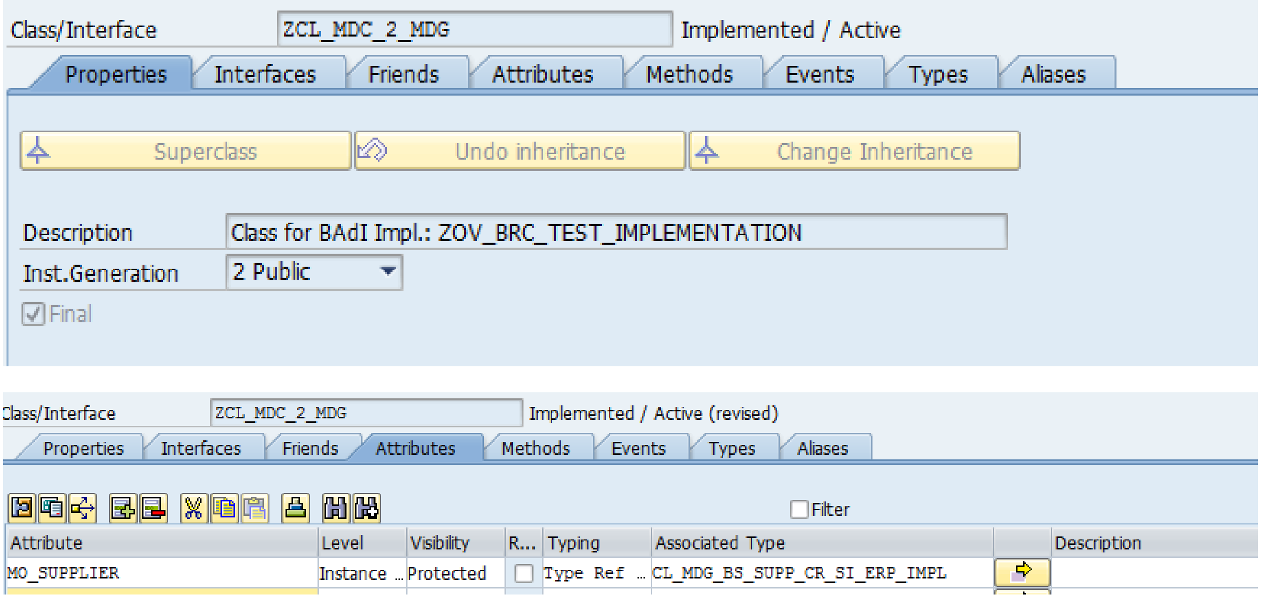
SAVE_BUSINESS_PARTNER_ACTIVE

```
  METHOD save_business_partner_active.
    DATA lt_cvi_ei_extern     TYPE cvis_ei_extern_t.
    DATA lv_parnter           TYPE bu_partner.
    DATA lr_fsbp_object       TYPE REF TO fsbp_bo_cvi.
    DATA lv_lifnr             TYPE lifnr.

    DATA lt_current_supp_rat TYPE TABLE OF zsupplier_rating.
    DATA lt_db_supp_rat       TYPE TABLE OF zsupplier_rating.
    DATA ls_db_supp_rat       TYPE zsupplier_rating.
    DATA ls_supp_rat_data     TYPE zsupplier_rating.
    DATA lt_supp_rat_ins      TYPE TABLE OF zsupplier_rating.
    DATA lt_supp_rat_upd      TYPE TABLE OF zsupplier_rating.
    DATA lt_supp_rat_del      TYPE TABLE OF zsupplier_rating.
    DATA lt_supp_rat_mod      TYPE TABLE OF zsupplier_rating.

    FIELD-SYMBOLS <ls_cvi_extern>     LIKE LINE OF mt_cvi_ei_extern.
    FIELD-SYMBOLS <ls_cvi_ext>        TYPE cvis_ei_extern.
    FIELD-SYMBOLS <ls_bp_keys>       LIKE LINE OF mt_bp_keys.
    FIELD-SYMBOLS <ls_supp_rat>      TYPE zei_supplier_rating.
    FIELD-SYMBOLS <ls_supp_rat_data> TYPE zsupplier_rating.
    FIELD-SYMBOLS <ls_supp_rat_db>   TYPE zsupplier_rating.

    super->save_business_partner_active( ).

    LOOP AT me->mt_cvi_ei_extern ASSIGNING <ls_cvi_extern>
                                     WHERE target = '1'.
      APPEND <ls_cvi_extern>-data TO lt_cvi_ei_extern.
    ENDLOOP.

    LOOP AT lt_cvi_ei_extern ASSIGNING <ls_cvi_ext>.
      CLEAR lv_parnter.

      READ TABLE me->mt_bp_keys ASSIGNING <ls_bp_keys>
       WITH KEY partner_guid = CONV bu_partner_guid( <ls_cvi_ext>-partner-header-
object_instance-bpartnerguid ).

      CHECK sy-subrc = 0.
      IF <ls_bp_keys>-partner IS INITIAL.
        CALL FUNCTION 'BUPA_NUMBERS_GET'
          EXPORTING
            iv_partner_guid = CONV bu_partner_guid( <ls_cvi_ext>-partner-header-
object_instance-bpartnerguid )
          IMPORTING
            ev_partner      = lv_parnter.
      ENDIF.

*     get standard supplier
      lr_fsbp_object ?= fsbp_business_factory=>get_instance( i_partner = lv_parnter )
.

      IF lr_fsbp_object IS BOUND.
        lv_lifnr = lr_fsbp_object->vendor->get_vendor( ).
        lv_kunnr = lr_fsbp_object->customer->get_customer( ).
      ENDIF.

*     get DB-state before update
      IF lv_lifnr IS NOT INITIAL.
        SELECT * FROM zsupplier_rating INTO TABLE lt_db_supp_rat
                                            WHERE lifnr = lv_lifnr.
      ENDIF.

*     current data
      LOOP AT <ls_cvi_ext>-vendor-central_data-rating-
zz_rating ASSIGNING <ls_supp_rat>.
        CLEAR ls_supp_rat_data.
        IF <ls_cvi_ext>-vendor-header-object_instance-lifnr IS INITIAL.
          ls_supp_rat_data-lifnr = lv_lifnr.
        ELSE.
          ls_supp_rat_data-lifnr = <ls_cvi_ext>-vendor-header-object_instance-lifnr.
        ENDIF.

        ls_supp_rat_data-zsupp_rat = <ls_supp_rat>-data_key-rating_id.
        ls_supp_rat_data-supp_rat = <ls_supp_rat>-data-rating.
        CASE <ls_supp_rat>-task.
          WHEN 'D'.
```

```
            APPEND ls_supp_rat_data TO lt_supp_rat_del.
          WHEN 'I'.
            APPEND ls_supp_rat_data TO lt_supp_rat_ins.
          WHEN 'U'.
            APPEND ls_supp_rat_data TO lt_supp_rat_upd.
          WHEN 'M'.
            APPEND ls_supp_rat_data TO lt_supp_rat_mod.
        ENDCASE.
      ENDLOOP.

      lt_current_supp_rat = lt_db_supp_rat.

*   merge
*   Deletes
      LOOP AT lt_supp_rat_del ASSIGNING <ls_supp_rat_data>.
        READ TABLE lt_current_supp_rat WITH KEY lifnr     = <ls_supp_rat_data>-lifnr
                                                zsupp_rat = <ls_supp_rat_data>-
zsupp_rat
                                      ASSIGNING <ls_supp_rat_db>.
        IF sy-subrc = 0.
*       delete record in target table
          DELETE lt_current_supp_rat INDEX sy-tabix.
        ELSE.
*       record doesn't exist on DB -
> must be a new one, keep data to be checked consistent
          READ TABLE lt_supp_rat_ins WITH KEY lifnr     = <ls_supp_rat_data>-lifnr
                                              zsupp_rat = <ls_supp_rat_data>-
zsupp_rat
                                    ASSIGNING <ls_supp_rat_db>.

          IF sy-subrc = 0.
            DELETE lt_current_supp_rat INDEX sy-tabix.
          ENDIF.
        ENDIF.
      ENDLOOP.

*   Inserts
      LOOP AT lt_supp_rat_ins ASSIGNING <ls_supp_rat_data>.
        READ TABLE lt_current_supp_rat WITH KEY lifnr     = <ls_supp_rat_data>-lifnr
                                                zsupp_rat = <ls_supp_rat_data>-
zsupp_rat
                                      ASSIGNING <ls_supp_rat_db>.
*     insert record into target table
        IF sy-subrc <> 0.
          INSERT <ls_supp_rat_data> INTO TABLE lt_current_supp_rat.
        ENDIF.
      ENDLOOP.

*   Updates
      LOOP AT lt_supp_rat_upd ASSIGNING <ls_supp_rat_data>.
        READ TABLE lt_current_supp_rat WITH KEY lifnr     = <ls_supp_rat_data>-lifnr
                                                zsupp_rat = <ls_supp_rat_data>-
zsupp_rat
                                      ASSIGNING <ls_supp_rat_db>.

*     insert record into target table
        IF sy-subrc = 0.
          <ls_supp_rat_db> = <ls_supp_rat_data>.
        ENDIF.
      ENDLOOP.

*   Modify
      LOOP AT lt_supp_rat_mod ASSIGNING <ls_supp_rat_data>.
        READ TABLE lt_current_supp_rat WITH KEY lifnr     = <ls_supp_rat_data>-lifnr
                                                zsupp_rat = <ls_supp_rat_data>-
zsupp_rat
                                      ASSIGNING <ls_supp_rat_db>.

*     insert record into target table
        IF sy-subrc = 0.
          <ls_supp_rat_db> = <ls_supp_rat_data>.
        ELSE.
          INSERT <ls_supp_rat_data> INTO TABLE lt_current_supp_rat.
        ENDIF.
      ENDLOOP.

      CALL FUNCTION 'Z_SUPP_RATING_UPDATE' IN UPDATE TASK
        TABLES
          x_supp_ratings = lt_current_supp_rat
          y_supp_ratings = lt_db_supp_rat.

  ENDMETHOD.
```

4.2.3. Sample BAdI Implementation class

ZCL_MDC_2_MDG



Class/Interface ZCL_MDC_2_MDG Implemented / Active

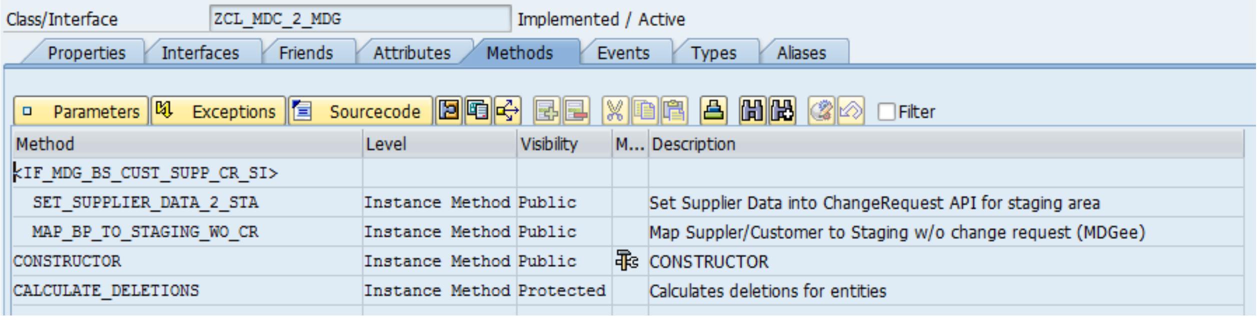
Description: Class for BAdI Impl.: ZOV_BRC_TEST_IMPLEMENTATION

Inst.Generation: 2 Public

☑ Final

Class/Interface ZCL_MDC_2_MDG Implemented / Active (revised)

Attribute	Level	Visibility	R...	Typing	Associated Type		Description
MO_SUPPLIER	Instance ...	Protected	□	Type Ref ...	CL_MDG_BS_SUPP_CR_SI_ERP_IMPL	⇨	



Method	Level	Visibility	M...	Description
<IF_MDG_BS_CUST_SUPP_CR_SI>				
SET_SUPPLIER_DATA_2_STA	Instance Method	Public		Set Supplier Data into ChangeRequest API for staging area
MAP_BP_TO_STAGING_WO_CR	Instance Method	Public		Map Suppler/Customer to Staging w/o change request (MDGee)
CONSTRUCTOR	Instance Method	Public		CONSTRUCTOR
CALCULATE_DELETIONS	Instance Method	Protected		Calculates deletions for entities

Method Implementations

CONSTRUCTOR

```
  METHOD constructor.
    CREATE OBJECT mo_supplier.
  ENDMETHOD.
```

CALCULATE_DELETIONS

```
METHOD calculate_deletions.
*** Read the existing active data via governance API. Compare the result
*** with the incoming data. Since the incoming data is current state,
*** existing data that is not part of the incoming data should be deleted.
  DATA:
    lt_return_messages TYPE bapirettab,
    lt_usmd_message    TYPE usmd_t_message,
    lv_index           TYPE i.

  FIELD-SYMBOLS:
    <ls_data>         TYPE any,
    <lv_assignment>   TYPE any,
    <lv_bp_header>    TYPE any,
    <lv_supp_rat_id> TYPE any.

  TRY.
*     read data
      ir_gov_api->read_entity(
        EXPORTING
          iv_entity_name    = iv_entity_name
          it_key            = it_key
          iv_crequest_id    = iv_crequest_id
          if_active_data    = abap_true
        IMPORTING
          et_data           = et_delete ) .

*     compare with incoming data
      IF et_delete IS NOT INITIAL.
        LOOP AT it_data ASSIGNING <ls_data>.
          CLEAR lv_index.
*         get keys
          ASSIGN COMPONENT:
            'BP_HEADER' OF STRUCTURE <ls_data> TO <lv_bp_header>,
            if_mdg_bs_ecc_bp_constants=>gc_ma_field-
assignment_id_mdg OF STRUCTURE <ls_data> TO <lv_assignment>,
            'ZSUPP_RAT' OF STRUCTURE <ls_data> TO <lv_supp_rat_id>.
          CHECK: <lv_bp_header> IS ASSIGNED.

*         read each entity with respect to keys only
          CASE iv_entity_name.
            WHEN 'ZSUPP_RAC'.
              CHECK: <lv_assignment> IS ASSIGNED,
                     <lv_supp_rat_id> IS ASSIGNED.

              READ TABLE et_delete TRANSPORTING NO FIELDS
                WITH KEY
                  (if_mdg_bs_ecc_bp_constants=>gc_ma_field-
assignment_id_mdg)  = <lv_assignment>
                  ('BP_HEADER')  = <lv_bp_header>
                  ('ZSUPP_RAT')  = <lv_supp_rat_id>.

              IF sy-subrc EQ 0.
                lv_index = sy-tabix.
              ENDIF.

            WHEN OTHERS.
              CONTINUE.
          ENDCASE.

*         cleanup
          UNASSIGN:  <lv_assignment>,
                     <lv_bp_header>,
                     <lv_bp_rat_id>,
                     <lv_cust_rat_id>,
                     <lv_supp_rat_id>.

*         delete
          CHECK lv_index NE 0.
          DELETE et_delete INDEX lv_index.
        ENDLOOP.
      ENDIF.

*   error handling
    CATCH cx_usmd_gov_api_core_error cx_usmd_gov_api.
      lt_usmd_message = ir_gov_api->get_messages( ).
      IF lt_usmd_message IS NOT INITIAL.
        CALL METHOD cl_mdg_bp_2_staging=>map_usmd_mess_to_bapiret
          EXPORTING
            it_usmd_message = lt_usmd_message
          IMPORTING
            et_bapirettab    = lt_return_messages.
        APPEND LINES OF lt_return_messages TO ct_return_messages.
      ENDIF.
  ENDTRY.
ENDMETHOD.
```

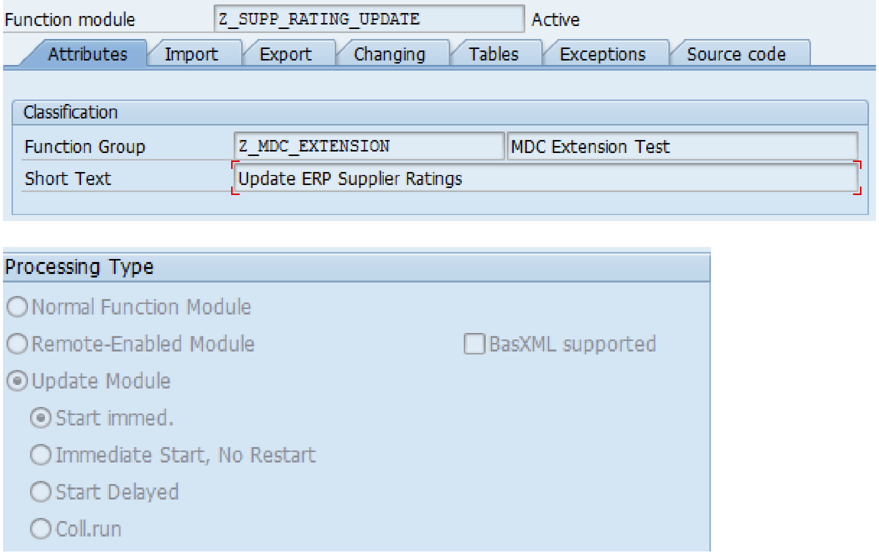
IF_MDG_BS_CUST_SUPP_CR_SI~SET_SUPPLIER_DATA_2_STA

```
  METHOD if_mdg_bs_cust_supp_cr_si~set_supplier_data_2_sta.
    DATA: ls_bp                  TYPE mdg_bs_bp_s_extern,
          lv_bp_header           TYPE bu_businesspartner,
          lv_bp_exists           TYPE abap_bool,
          ls_return_message  TYPE bapiret2,
          lt_return_messages TYPE bapirettab,
          lt_usmd_message    TYPE usmd_t_message.

    DATA: ls_sta_zsupp_rac TYPE zzoli_s_bp_pp_zsupp_rac,
          lt_sta_zsupp_rac TYPE STANDARD TABLE OF zzoli_s_bp_pp_zsupp_rac.

    DATA: lt_del_zsupp_rac TYPE STANDARD TABLE OF zzoli_s_bp_pp_zsupp_rac,
          lt_key_zsupp_rac TYPE STANDARD TABLE OF zzoli_s_bp_pp_zsupp_rac.

    DATA: lt_entities TYPE STANDARD TABLE OF usmd_entity,
          lv_entity   TYPE usmd_entity.

    FIELD-SYMBOLS: <lt_data>   TYPE ANY TABLE,
                   <lt_delete> TYPE ANY TABLE.

    FIELD-SYMBOLS: <ls_supp_rating> TYPE zei_supplier_rating.

    mo_supplier->if_mdg_bs_cust_supp_cr_si~set_supplier_data_2_sta(
      EXPORTING is_bp_api          = is_bp_api
                ir_cr_api          = ir_cr_api
                ir_gov_api         = ir_gov_api
                iv_crequest_id     = iv_crequest_id
                iv_temp_sta_bp_id  = iv_temp_sta_bp_id
       CHANGING ct_return_messages = ct_return_messages ).

*** APIs MUST be provided, otherwise the processing doesn't make sense.
    IF NOT ir_gov_api IS BOUND.
*   should never happen at this point in time
      ls_return_message-type   = 'E'.
      ls_return_message-id     = 'MDG_BS_BP_DATAREPL'.
      ls_return_message-number = '10'.
      APPEND ls_return_message TO ct_return_messages.
      RETURN.
    ENDIF.

* 1) move input into local structure
    ls_bp = is_bp_api.

    IF ls_bp-partner-header-object_instance-bpartner IS NOT INITIAL.
      lv_bp_header = ls_bp-partner-header-object_instance-bpartner.

      CALL FUNCTION 'BUPA_EXISTENCE_CHECK'
        EXPORTING
          iv_partner = ls_bp-partner-header-object_instance-bpartner
        TABLES
          et_return  = lt_return_messages.

      LOOP AT lt_return_messages TRANSPORTING NO FIELDS WHERE type CA 'EAX'.
        lv_bp_exists = abap_false.
        EXIT.
      ENDLOOP.

      IF sy-subrc NE 0.
        lv_bp_exists = abap_true.
      ENDIF.
    ELSEIF iv_temp_sta_bp_id IS NOT INITIAL.
      lv_bp_header = iv_temp_sta_bp_id.
      lv_bp_exists = abap_false.
    ELSE.
      lv_bp_exists = abap_false.
    ENDIF.

* 2) map data
*   supplier rating
    LOOP AT ls_bp-vendor-central_data-rating-
zz_rating ASSIGNING <ls_supp_rating>.
      ls_sta_zsupp_rac-bp_header = lv_bp_header.
      ls_sta_zsupp_rac-
assgnm_id = if_mdg_bs_ecc_bp_constants=>gc_sp_std_mlt_assignment_id.
      ls_sta_zsupp_rac-zsupp_rat = <ls_supp_rating>-data_key-rating_id.
      ls_sta_zsupp_rac-supp_rat = <ls_supp_rating>-data-rating.

      APPEND ls_sta_zsupp_rac TO lt_sta_zsupp_rac.
    ENDLOOP.
*   prepare key for reading all
    IF ls_bp-vendor-central_data-rating-zzcurrent_state EQ abap_true
     AND lv_bp_exists EQ abap_true.
      CLEAR ls_sta_zsupp_rac.
      ls_sta_zsupp_rac-bp_header = lv_bp_header.
      ls_sta_zsupp_rac-
assgnm_id = if_mdg_bs_ecc_bp_constants=>gc_sp_std_mlt_assignment_id.
      APPEND ls_sta_zsupp_rac TO lt_key_zsupp_rac.
    ENDIF.


*** Check for deletions (starting with MDG 6.1). It might be possible that
*** some segments are deleted on the client. Keys to be used are the multiple
*** assignment keys!
    IF cl_mdg_bs_cust_switch_check=>mdg_bs_ecc_cust_switch_61( ) EQ abap_true.

*     supplier rating
      IF lt_key_zsupp_rac IS NOT INITIAL.
        me->calculate_deletions(
          EXPORTING
            ir_gov_api         = ir_gov_api
            it_data            = lt_sta_zsupp_rac
            it_key             = lt_key_zsupp_rac
            iv_crequest_id     = iv_crequest_id
            iv_entity_name     = 'ZSUPP_RAC'
          IMPORTING
            et_delete          = lt_del_zsupp_rac
          CHANGING
            ct_return_messages = ct_return_messages ).
      ENDIF.
    ENDIF.

*** Writing a supplier to staging with the governance API consists of
*** writing new or changed data and deleting deleted data. The executed API
*** methods are always identical. Therefore the processing can be done in
*** a loop over the entities that actually have data to be written / deleted.
** Determine entities to handle

* check: supplier rating
    IF lt_sta_zsupp_rac[] IS NOT INITIAL
     OR lt_del_zsupp_rac[] IS NOT INITIAL.
      APPEND 'ZSUPP_RAC' TO lt_entities.
    ENDIF.

** Loop over the collected entities to write or delete in staging.
    LOOP AT lt_entities INTO lv_entity.
      TRY.
*       1. prepare the data
          UNASSIGN: <lt_data>, <lt_delete>.
          CASE lv_entity.
            WHEN 'ZSUPP_RAC'.
              ASSIGN lt_sta_zsupp_rac TO <lt_data>.
              ASSIGN lt_del_zsupp_rac TO <lt_delete>.
            WHEN OTHERS.
          ENDCASE.

*         2. delete data reference in staging
          IF <lt_delete> IS ASSIGNED
            AND <lt_delete> IS NOT INITIAL.
            CALL METHOD ir_gov_api->delete_entity
              EXPORTING
                iv_crequest_id = iv_crequest_id
                iv_entity_name = lv_entity
                it_data        = <lt_delete>.
          ENDIF.

*         3. write data reference to staging
          IF <lt_data> IS ASSIGNED
            AND <lt_data> IS NOT INITIAL.
            CALL METHOD ir_gov_api->write_entity
              EXPORTING
                iv_crequest_id = iv_crequest_id
                iv_entity_name = lv_entity
                it_data        = <lt_data>.
          ENDIF.

*     error handling
        CATCH cx_usmd_gov_api_core_error cx_usmd_gov_api.
          lt_usmd_message = ir_gov_api->get_messages( ).
          IF lt_usmd_message[] IS NOT INITIAL.
            CALL METHOD cl_mdg_bp_2_staging=>map_usmd_mess_to_bapiret
              EXPORTING
                it_usmd_message = lt_usmd_message
              IMPORTING
                et_bapirettab   = lt_return_messages.
            APPEND LINES OF lt_return_messages TO ct_return_messages.
            CLEAR lv_entity.
            REFRESH: lt_usmd_message, lt_return_messages.
          ENDIF.
      ENDTRY.
    ENDLOOP. " end handle entities

  ENDMETHOD.
```

4.2.4. Sample Update function module

To update the customer table ZSUPPLIER_RATING function module Z_SUPP_RATING_UPDATE is used in this example.

Z_SUPP_RATING_UPDATE



Function module Z_SUPP_RATING_UPDATE Active

Classification

Function Group Z_MDC_EXTENSION MDC Extension Test

Short Text Update ERP Supplier Ratings

Processing Type

○ Normal Function Module

○ Remote-Enabled Module ☐ BasXML supported

● Update Module

- ● Start immed.
- ○ Immediate Start, No Restart
- ○ Start Delayed
- ○ Coll.run

```
FUNCTION Z_SUPP_RATING_UPDATE.
*"----------------------------------------------------------------------
*"*"Update Function Module:
*"
*"*"Local Interface:
*"  TABLES
*"      X_SUPP_RATINGS STRUCTURE  ZSUPPLIER_RATING
*"      Y_SUPP_RATINGS STRUCTURE  ZSUPPLIER_RATING
*"----------------------------------------------------------------------

  DATA: ls_y_supp_rating TYPE zsupplier_rating.

  LOOP AT y_supp_ratings INTO ls_y_supp_rating.
    READ TABLE x_supp_ratings WITH KEY lifnr          = ls_y_supp_rating-lifnr
                                       zsupp_rat      = ls_y_supp_rating-
zsupp_rat.

    IF sy-subrc <> 0.
      DELETE FROM zsupplier_rating WHERE lifnr       = ls_y_supp_rating-lifnr
                                     AND zsupp_rat   = ls_y_supp_rating-
zsupp_rat.
      IF sy-subrc <> 0.
        MESSAGE x000(zcustomer_rating).
      ENDIF.
    ENDIF.
  ENDLOOP.

  MODIFY zsupplier_rating FROM TABLE x_supp_ratings.  "Current State
  IF sy-subrc <> 0.
    MESSAGE x000(zcustomer_rating).
  ENDIF.

ENDFUNCTION.
```

5. ADDITIONAL INFORMATION

5.1. Further Reading

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
- Learn more: Latest Release | Webinars | Help Portal | How-to Information | Key Presentations

SAP Roadmap Explorer

- Please see the roadmap for SAP Master Data Governance

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

5.2. SAP Notes

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

Note Number	Note Description
2221398	MDG-BP/C/S/CA: (Un-)Supported Fields in Data Model BP
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
3194967	MDG Customer Connection 2021 for S/4HANA 2022
3043582	MDG Customer Connection 2020
3134600	MDG-M: Supported fields in Data Model MM

1806108	Functional restrictions in MDG-M in MDG7 (incl. SP02)
2129261	Functional restrictions in MDG-M in MDG8
2284745	Functional Restrictions in MDG for Material with SAP Master Data Governance 9.0
2461516	Functional Restrictions in MDG for Material with SAP Master Data Governance 9.1
2656693	Functional Restrictions in MDG for Material in SAP Master Data Governance 9.2 and on SAP S/4HANA 1809
2816571	Functional Restrictions in MDG for Material on SAP S/4HANA 1909
2948873	Functional Restrictions in MDG for Material on SAP S/4HANA 2020
3070012	Functional Restrictions in MDG for Material on SAP S/4HANA 2021
3219945	Functional Restrictions in MDG for Material on SAP S/4HANA 2022
2479869	Usage of Lean Classification with SAP Master Data Governance
1619534	How to Create, Enhance and Adapt FPM Applications

www.sap.com/contactsap

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

THE BEST RUN SAP®