PUBLIC

How-To: Node (Table) Extension for Material/Product in MDG Consolidation and Mass Processing for Releases Starting with SAP S/4HANA 2020

Applicable Releases:

SAP MDG, Consolidation and SAP MDG, Mass Processing running on SAP S/4HANA 2020 or higher.

As of S/4HANA 2020 there was a change in the API for Material/Product, therefore the extensibility guide must be updated as well (main reason: new BAdI must be used)

Version 1.1

April 2024

Document History

Document Version	Description
1.0	First official release of this guide
1.1	New document template (no content changes)

1 Business Scenario

SAP MDG, Consolidation and SAP MDG, Mass Processing are applications that provide optimized Fiori-based mass transactions. If the domain-specific solution does not fully meet customers' requirements, these applications can be customized and extended.

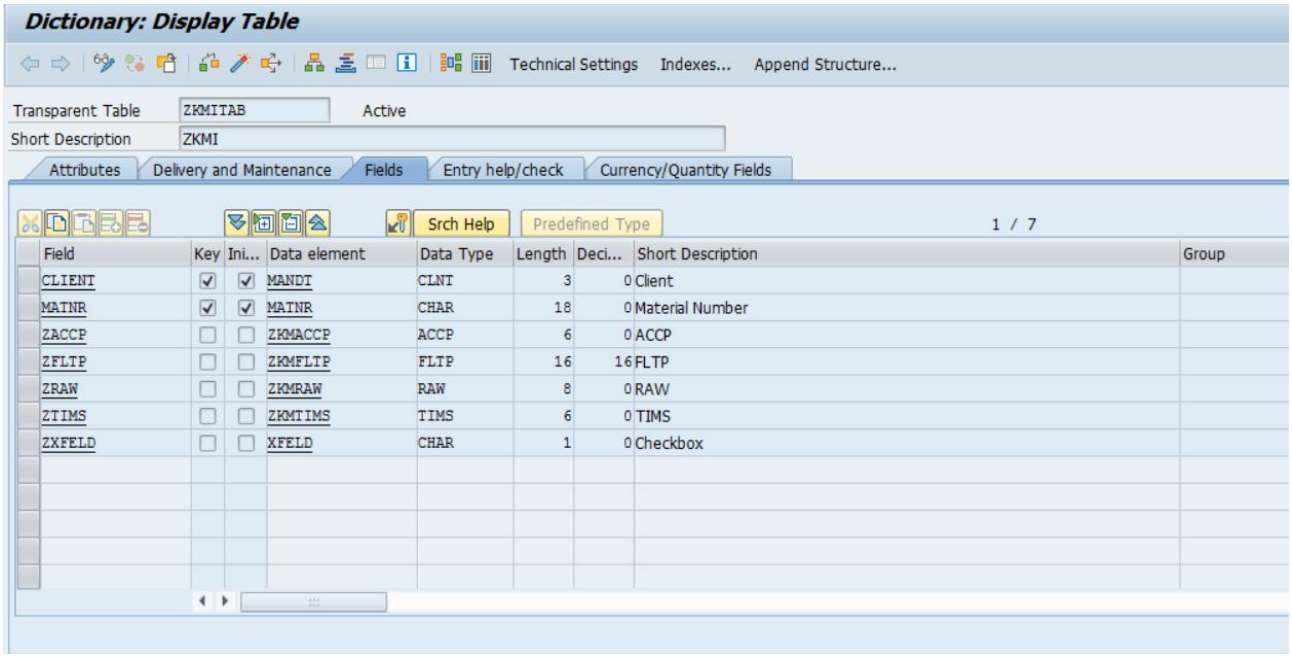
This guide explains how to extend the Material/Product data model in SAP MDG, Consolidation and SAP MDG, Mass Processing for a new node (table). Only the part inside Consolidation and Mass Processing will be described. The extension for the active table in the backend has been already done.

We recommend studying the following How-To Guides before working with this one that describes the backend extensions:

SAP How-To Guide: Extend MDG-M Data Model by a New Entity Type (ERP Table, Reuse Option)

SAP How-To Guide: Extend MDG-M Data Model by a New Entity Type (Reuse Option)

In this example, an additional material table ZKMITAB was created in the backend that refers to a material (MATNR).



2 Functional Restrictions

MDG flex data models are not supported. This guide describes only extensions for tables in customer namespace.

3 Step by Step Explanation

The following steps provide details on how to extend MDG, Consolidation or MDG, Mass Processing by adding a new table ZKMITAB for ERP Material.

MDG, Consolidation and MDG, Mass Processing support two different scenarios:

1. Central Governance **and** Consolidation/Mass Processing
   Implement the extensions in Central Governance (MDG) as described in the SAP MDG How-To Guide
   Extend MDG Material – Node Extension (Reuse Option) for custom tables
2. Consolidation/Mass Processing **only**
   If a consolidation or mass process runs without MDG change requests, it is sufficient to follow the SAP MDG How-To Guide:
   a. To enhance the data-Structure and x-Structure for the customer table in the structure CMD_BS_MAT_S_MAT_DATA. Also do the same for CMD_PRD_S_UNIFIED_PROD_DATA if you are on S/4 HANA 2020 and higher.
   b. To implement the extension for SAP BAdI CMD_BS_MAT_API_SEGMENTS_EXT to check and save customer table data. For details, see chapter 2 in this document.

For both scenarios, the additional steps to be done in Consolidation/Mass Processing are described below.

3.1 Redefine Material data access class

To use and access data in the newly created source and process tables, the following class must be created. It must inherit from the given superclass and certain methods must be redefined.

A detailed source code example is provided in chapter 4.11 in the appendix section.

As already stated, if the processed data will be passed to a Change Request, the corresponding MDG extensions must have been implemented. They are not part of this guide.

Create and redefine Material data access class

Create the new class ZCL_MDC_DATA_MAT that inherits from the class CL_MDC_DATA_MAT. The new class will then be used in consolidation processes involving the Process Model 194 (Material Data). Note that for all customer extensions of the Process Model 194 exactly this new Z-class must be used.

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. Otherwise, new source and process tables can automatically be created with the suffixes _SRC and _PRC. This method can also be redefined to use the custom suffixes or prefixes.
- APPEND_ACTIVE_RECORDS
  A redefinition of this method is only required if the active data does not map natively to the extended consolidation model (move-corresponding).

Fields using a “Large Object Binary” data type

If your custom field uses a “Large Object Binary” related data type (for example a string, blob, raw binary or similar), you need to redefine one more method in your custom Product 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 Object Binary” data type. In this case, ensure that the method returns “abap_true” for the affected tables.

3.2 Implement BAdI CMD_PRODUCT_SEGMENTS_EXT

Example BAdI Implementation for CMD_PRODUCT_SEGMENTS_EXT

1. CHECK Method
   This method is used to perform the validation for custom data. In case of an error, an error message is returned with et_message variable to stop the further processing
   In case of a successful check or no errors, the data is saved to class attributes which will be used later in the SAVE method as the SAVE method does not have any input parameters.

```
  METHOD if_cmd_product_segments_ext~check.
    DATA: ls_message LIKE LINE OF et_messages.
    LOOP AT it_data ASSIGNING FIELD-SYMBOL(<ls_data>).
      INSERT LINES OF <ls_data>-ymdgm_bupa01_tab INTO TABLE
mt_mdgm_bupa01_modify.
    ENDLOOP.
    LOOP AT mt_mdgm_bupa01_modify ASSIGNING FIELD-
SYMBOL(<ls_mdgm_bupa01_modify>) WHERE nickname IS INITIAL.
      ls_message-product = <ls_mdgm_bupa01_modify>-matnr.
      ls_message-msgty = 'E'.
      ls_message-msgid = 'MG'.
      ls_message-msgno = '899'.
      ls_message-msgv1 = 'Please enter a valid value for the nick name'
##NO_TEXT.
      INSERT ls_message INTO TABLE et_messages.
      DELETE mt_mdgm_bupa01_modify WHERE matnr = <ls_mdgm_bupa01_modify>-
matnr.
    ENDLOOP.
    LOOP AT mt_mdgm_bupa01_modify ASSIGNING FIELD-SYMBOL(<ls_mod>) WHERE
delete_row IS NOT INITIAL.
      INSERT <ls_mod> INTO TABLE mt_mdgm_bupa01_del.
      DELETE mt_mdgm_bupa01_modify WHERE matnr = <ls_mod>-matnr AND
bupa_id = <ls_mod>-bupa_id.
    ENDLOOP.

  ENDMETHOD.
```

2. SAVE Method
   This method copies the data which has been saved in the buffers or class attributes during the check to the database tables.

```
METHOD if_cmd_product_segments_ext~save.
    DATA: lr_cx_cmd_prod_maint_api TYPE REF TO cx_cmd_product_maint_api,
          lr_exception             TYPE REF TO cx_cmd_product_maint_api,
          lv_subrc                 TYPE sy-subrc VALUE IS INITIAL.


    IF mt_mdgm_bupa01_del IS NOT INITIAL.
      DELETE ymdgm_bupa00 FROM TABLE mt_mdgm_bupa01_del.
      IF sy-subrc <> 0.
```

```
        lv_subrc = sy-subrc.
      ENDIF.
    ENDIF.

    IF mt_mdgm_bupa01_modify IS NOT INITIAL.
      MODIFY ymdgm_bupa00 FROM TABLE mt_mdgm_bupa01_modify.
      IF sy-subrc <> 0.
        lv_subrc = sy-subrc.
      ENDIF.
    ENDIF.


    IF lv_subrc <> 0.
      CLEAR lr_cx_cmd_prod_maint_api.
      CREATE OBJECT lr_cx_cmd_prod_maint_api
        EXPORTING
          previous = lr_exception.
      IF lr_cx_cmd_prod_maint_api IS BOUND.
        lr_cx_cmd_prod_maint_api->iv_matnr                      = ''.
        lr_cx_cmd_prod_maint_api->if_t100_dyn_msg~msgty         = 'E'.
        lr_cx_cmd_prod_maint_api->if_t100_message~t100key-msgid = '00'.
        lr_cx_cmd_prod_maint_api->if_t100_message~t100key-msgno = '000'.
        lr_cx_cmd_prod_maint_api->if_t100_message~t100key-attr1 = 'Error
in the record: '.
        lr_cx_cmd_prod_maint_api->if_t100_message~t100key-attr2 = ''.
        lr_cx_cmd_prod_maint_api->if_t100_message~t100key-attr3 = ''.
        lr_exception = lr_cx_cmd_prod_maint_api.
      ENDIF.
    ENDIF.

    IF lr_exception IS BOUND.
      RAISE EXCEPTION lr_exception.
    ENDIF.

  ENDMETHOD.
```

3. GET_PRODUCT_ALIAS
   This method provides the mapping information about the MATNR field in a custom table. If the field MATNR exists, it can be used directly. However, if there is a field with a different name and the field represents MATNR, then the field name should be provided here.

```
  METHOD if_cmd_product_segments_ext~get_product_alias.
    DATA ls_tab_map TYPE if_cmd_product_segments_ext~ty_s_product_alias.
    ls_tab_map-tabname = 'YMDGM_BUPA01_TAB'.
    ls_tab_map-fieldname = 'MATNR'.
    INSERT ls_tab_map INTO TABLE et_product_alias_mapping.
    ls_tab_map-tabname = ' YMDGM_BUPA01_X_TAB '.
    ls_tab_map-fieldname = 'MATNR'.
    INSERT ls_tab_map INTO TABLE et_product_alias_mapping.
  ENDMETHOD.
```

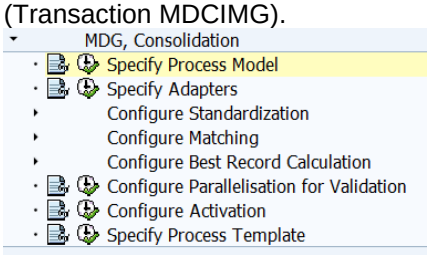
4. CLEAN_UP
   This method is used to clear the class attributes of the BAdI implementation class. A cleanup is done along with other standard cleanup activities by the API.

```
  METHOD IF_CMD_PRODUCT_SEGMENTS_EXT~CLEAN_UP.
    CLEAR mt_mdgm_bupa01_modify.
    CLEAR mt_mdgm_bupa01_del.
  ENDMETHOD.
```

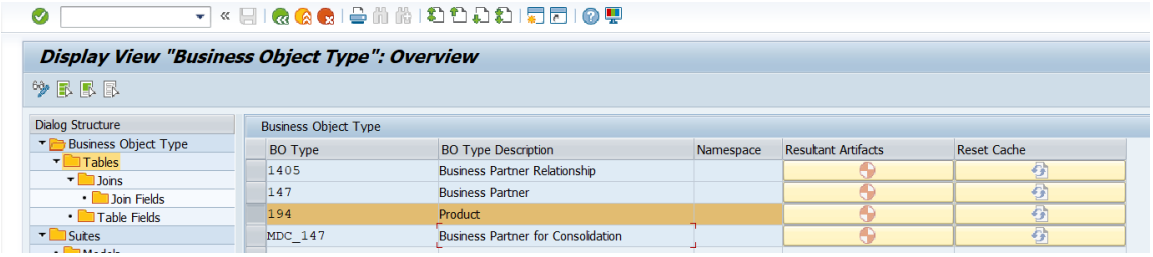
3.3 Extend Process Model

To extend the Process Model with a new table, the contents of the 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.

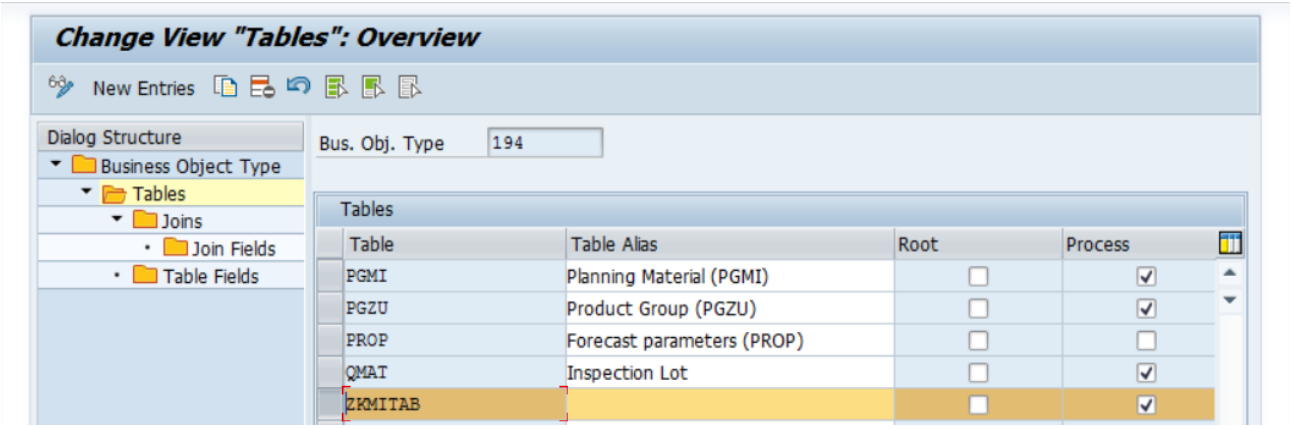
1. Start the IMG Activity *Specify Process Model* in the IMG for Consolidation and Mass Processing (Transaction MDCIMG).



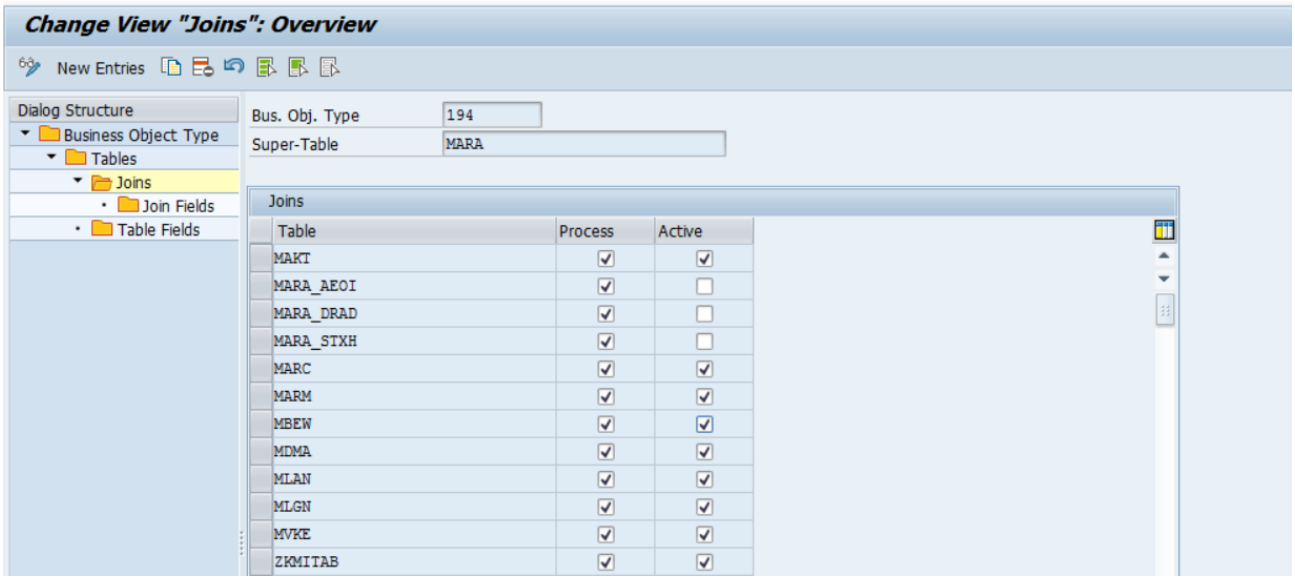
2. Select the Business Object Type 194 (Material) and navigate to the sub-node Tables.



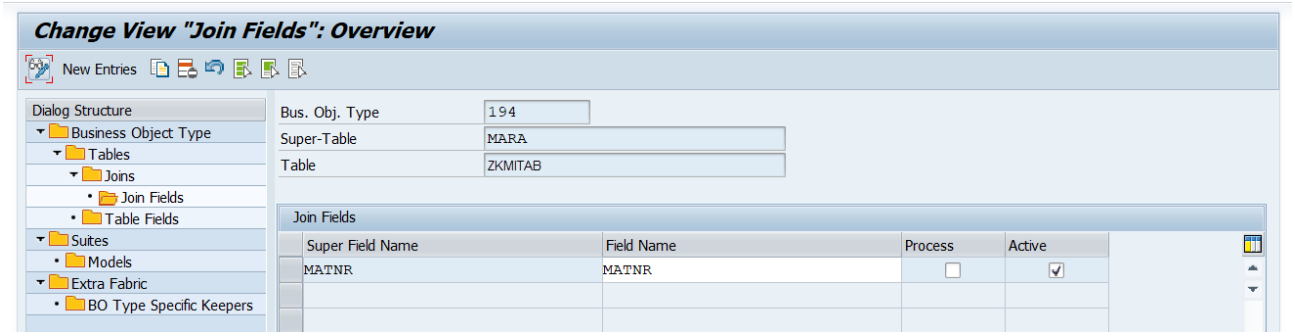
Add a new process table ZKMITAB to the process model. Add the name of the database table to the *Table* column and mark the *Process* column.



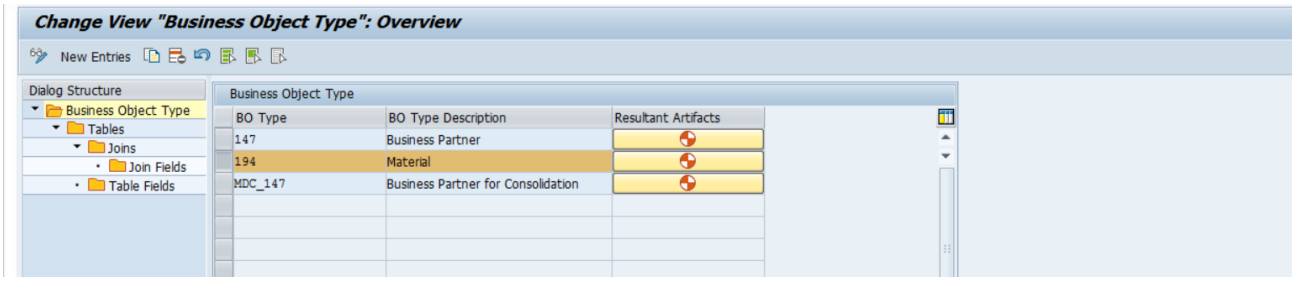
3. Save the changes.
4. Add a new join to the table MARA by selecting the table line and navigating into Joins. Add the new entry ZKMITAB and mark *Process* and *Active*.



5. Save the changes.
6. Select the added Join and navigate into the Join Fields in order to link the parent table and child



7. Save the changes.
8. Generate artifacts for the new table ZKMITAB for the Business Object Type 194.



9. Select all rows containing the new table name ZKMITAB, choose a package in which the new objects shall be created, and choose *Apply Selected*. (Alternatively, *Apply Missing* can be used.)

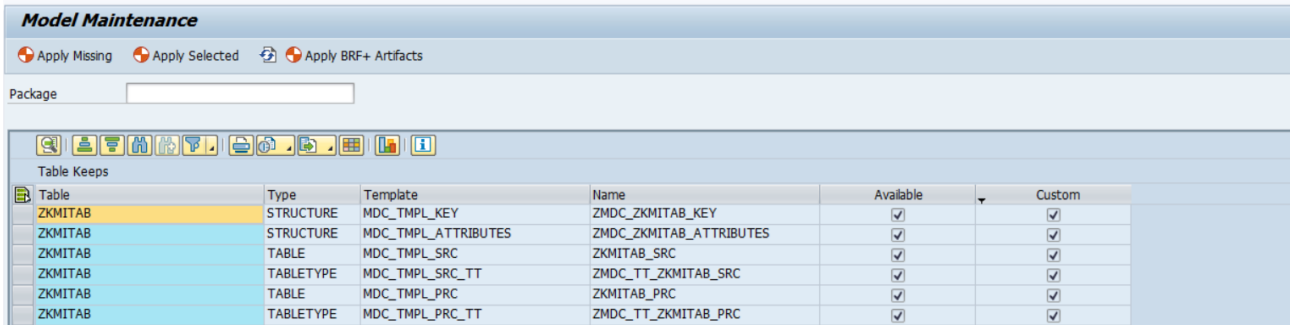
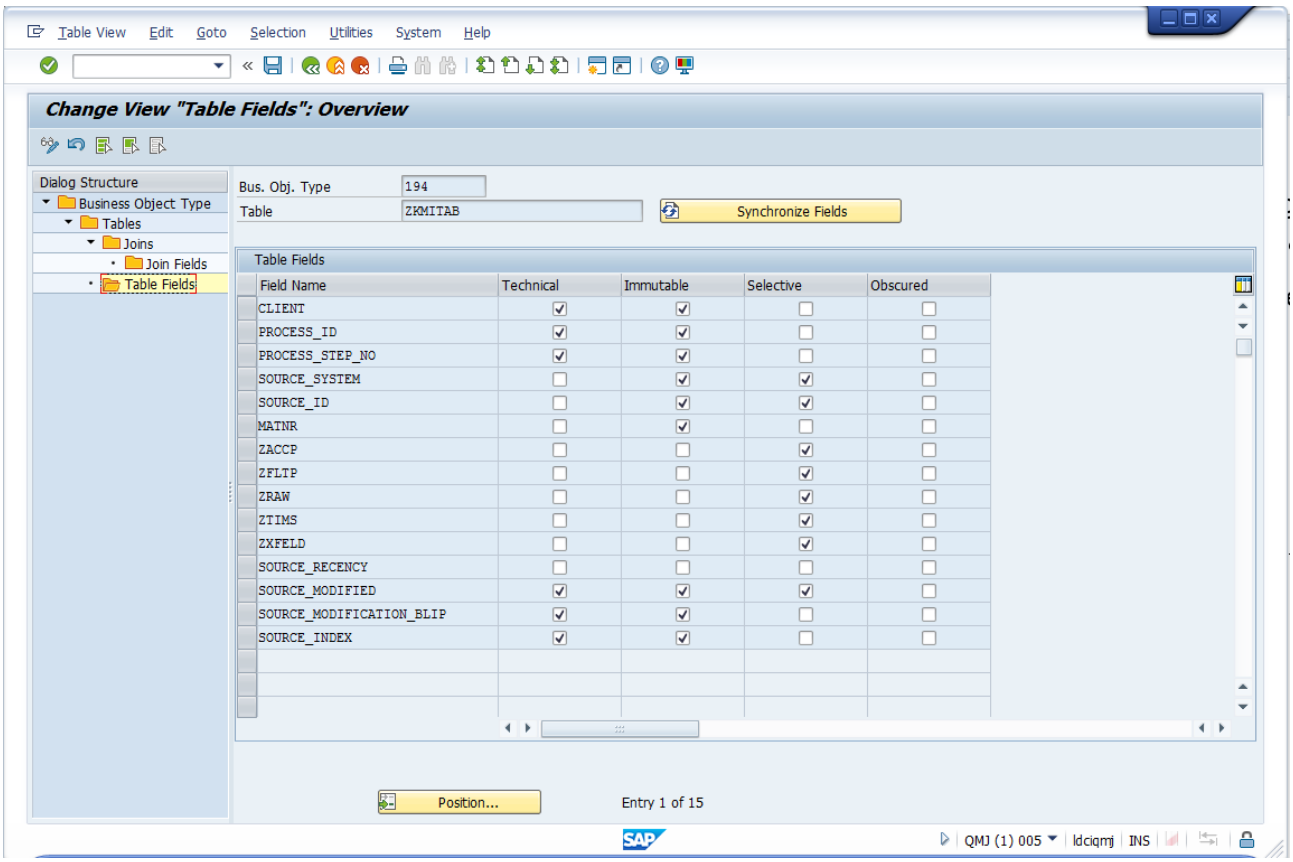


Table	Type	Template	Name	Available	Custom
ZKMITAB	STRUCTURE	MDC_TMPL_KEY	ZMDC_ZKMITAB_KEY	✓	✓
ZKMITAB	STRUCTURE	MDC_TMPL_ATTRIBUTES	ZMDC_ZKMITAB_ATTRIBUTES	✓	✓
ZKMITAB	TABLE	MDC_TMPL_SRC	ZKMITAB_SRC	✓	✓
ZKMITAB	TABLETYPE	MDC_TMPL_SRC_TT	ZMDC_TT_ZKMITAB_SRC	✓	✓
ZKMITAB	TABLE	MDC_TMPL_PRC	ZKMITAB_PRC	✓	✓
ZKMITAB	TABLETYPE	MDC_TMPL_PRC_TT	ZMDC_TT_ZKMITAB_PRC	✓	✓

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

- Source database table (ZKMITAB_SRC) & corresponding table type
- Process database table (ZKMITAB_PRC) & corresponding table type
- Key and attribute structures

10. Synchronize *Table Fields* and define your settings.



Field Name	Technical	Immutable	Selective	Obscured
CLIENT	✓	✓		
PROCESS_ID	✓	✓		
PROCESS_STEP_NO	✓	✓		
SOURCE_SYSTEM		✓	✓	
SOURCE_ID		✓	✓	
MATNR		✓		
ZACCP			✓	
ZFLTP			✓	
ZRAW			✓	
ZTIMS			✓	
ZXFELD			✓	
SOURCE_RECENCY				
SOURCE_MODIFIED	✓	✓	✓	
SOURCE_MODIFICATION_BLIP	✓	✓		
SOURCE_INDEX	✓	✓		

3.4 Redefine Material model class

To use and access data in the newly created source and process tables within a process, the following class must be created. It must inherit from the given superclass and certain methods must be redefined.

A detailed source code example is provided in the Appendix section.

Also note that, as already stated, if the processed data will be passed to a Change Request, the corresponding MDG extensions must have been implemented. They are not part of this guide.

Create and redefine Material model implementation class

Create a new class ZCL_MDC_MODEL_MAT that inherits from the class CL_MDC_MODEL_MAT. The new class will then be used in processes involving the Process Model 194 (Material Data). Note that for all material extensions of the Process Model 194, exactly this new Z-class must be used.
The following methods must be redefined:

- **READ_ALL_DATA**
- **MAP_EXTENSIONS_2API**
- **CALL_API_EXTENSION_PREPARE**
- **MAP_EXT_TABLE_2_UNI** This method enables you to delete table entries. It is provided with the SAP Note 3196266 (CMP Material Master: Enable Deletions for node extensibility).

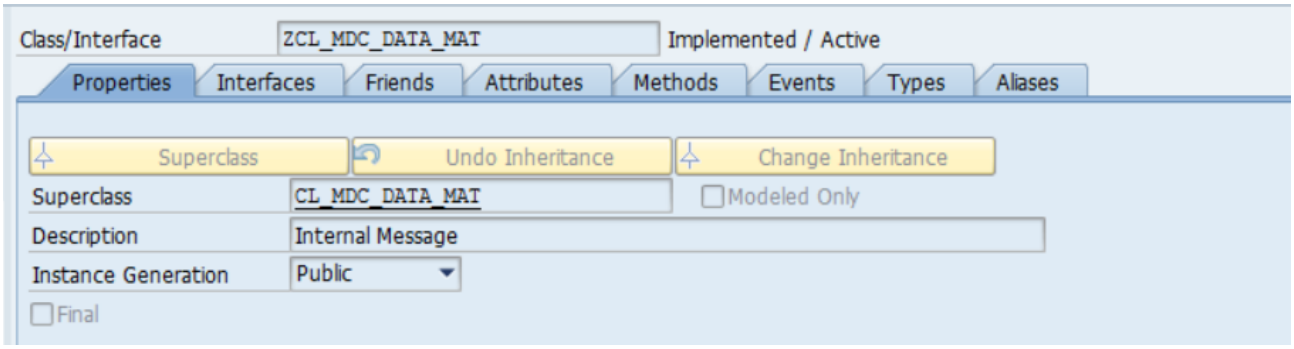
In addition, an object of this class will only be instantiable by a new method of the class CL_MDC_MODEL. Therefore, the *Instance Generation* must be set to *Protected* and the interface IF_MDC_MODEL must be maintained.

4 Additional Information

4.1 Sample Source Code

4.1.1 Sample source code data access class

The customer data access class for Material must inherit from CL_MDC_DATA_MAT.



Method Redefinitions

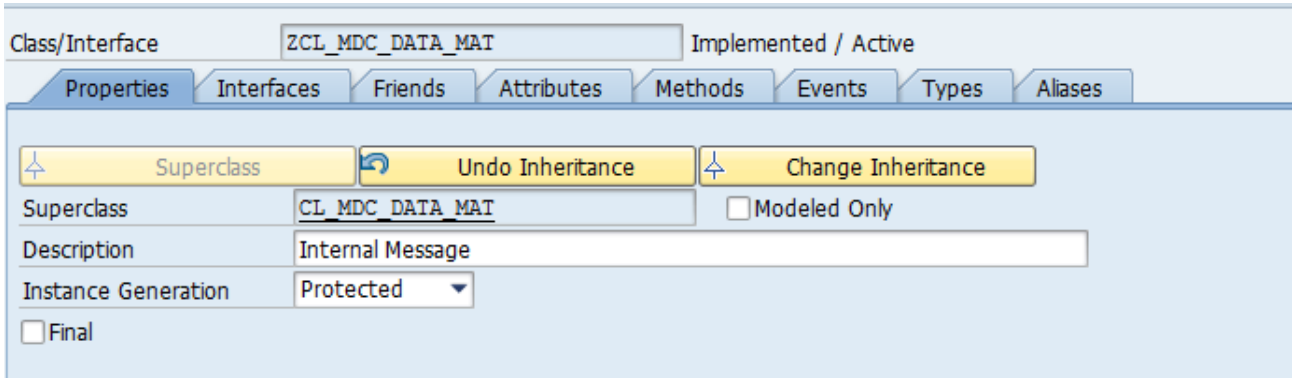
In the given example for the table ZKMITAB no methods need to be redefined, so the table natively maps via MATNR (and other attributes) to active data.

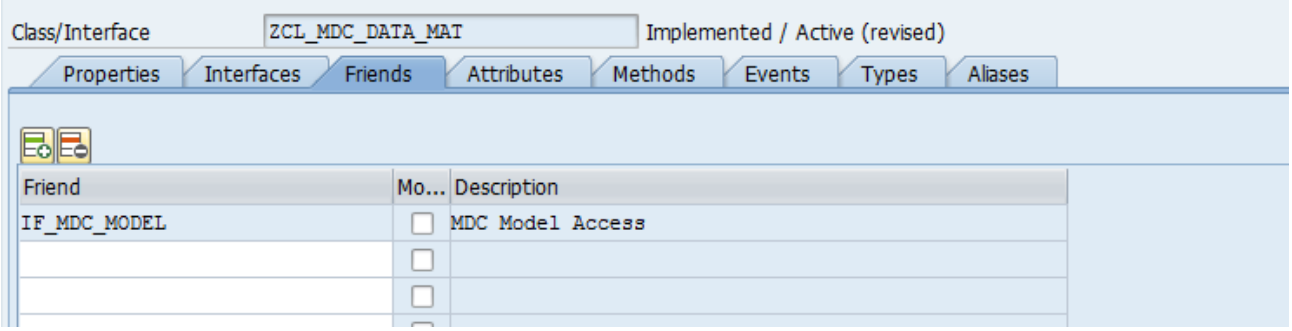
For redefinition examples see the following guide:

SAP How-To Guide: Extend the MDG, Consolidation and Mass Processing Business Partner/Supplier/Customer - Field Extension

4.1.2 Sample source code model implementation class

The customer model implementation class for Material must inherit from CL_MDC_MODEL_MAT.





Class/Interface ZCL_MDC_MODEL_MAT Implemented / Active

Properties | Interfaces | Friends | Attributes | Methods | Events | Types | Aliases

Properties Filter

Attribute	Level	Visibility	R...	Typing	Associated Type	
MR_MLGT_PRC	Instance Attribute	Protected		Type Ref To	MDC_TT_MLGT_PRC	Lis
MR_MVKE_PRC	Instance Attribute	Protected		Type Ref To	MDC_TT_MVKE_PRC	Lis
MR_MVKE_STXH_PRC	Instance Attribute	Protected		Type Ref To	MDC_TT_MVKE_STXH_PRC	Lis
MR_MVKE_STXL_PRC	Instance Attribute	Protected		Type Ref To	MDC_TT_MVKE_STXL_PRC	Lis
MR_PGMI_PRC	Instance Attribute	Protected		Type Ref To	MDC_TT_MAT_PGMI_PRC	Lis
MR_PGZU_PRC	Instance Attribute	Protected		Type Ref To	MDC_TT_MAT_PGZU_PRC	Lis
MR_QMAT_PRC	Instance Attribute	Protected		Type Ref To	MDC_TT_QMAT_PRC	Lis
MT_MARA_SRC_STAT_UPD	Instance Attribute	Protected		Type	MDC_TT_MARA_SRC_STAT	Lis
MT_MATCH_GROUPS	Instance Attribute	Protected		Type	MDC_TT_MATCH_GROUP	Ma
MT_MAT_KEYS	Instance Attribute	Protected		Type	MDC_TT_MAT_KEYS	Ma
MT_MDC_MAT_S_MAT_DATA	Instance Attribute	Protected		Type	MDC_TT_MAT_S_MAT_DATA	Ma
MV_ACTIVATION	Instance Attribute	Protected		Type	BOOLE_D	Ac
MV_MAT_ID_TEMP_PREFIX	Instance Attribute	Protected		Type	CHAR2	Ma
MV_PARALLEL	Instance Attribute	Protected		Type	ABAP_BOOL	
MT_MDC_KEY_MAP	Instance Attribute	Protected		Type	TY_TT_MDC_KEY_MAP	
MR_ZMARA_KSSK_PRC	Instance Attribute	Private		Type Ref To	ZMDC_TT_ZMARA_KSSK_PRC	Lis
MR_ZMARA_AUSP_PRC	Instance Attribute	Private		Type Ref To	ZMDC_TT_ZMARA_AUSP_PRC	Lis
MR_ZKMITAB_PRC	Instance Attribute	Private		Type Ref To	ZMDC_TT_ZKMITAB_PRC	Lis

Method Redefinitions

READ_ALL_DATA

```
METHOD read_all_data.

  CHECK me->mr_mara_prc IS NOT BOUND.
  super->read_all_data( it_source_keys = it_source_keys iv_package_number = iv_package_number ).

  me->mr_zkmitab_prc    =
  CAST #( me->object( 'ZKMITAB' )->read( it_source_keys = it_source_keys  iv_package_number = iv_package_number ) ).

ENDMETHOD.
```

MAP_EXTENSIONS_2API

```
METHOD map_extensions_2api.

  FIELD-SYMBOLS:
    <prc>          TYPE any.
  DATA:
     ls_zkmi        TYPE zkmi.

  SORT me->mr_zkmitab_prc->* BY process_id process_step_no source_system source_id.

  READ TABLE me->mr_zkmitab_prc->* ASSIGNING <prc>
    WITH KEY
      process_id        = is_mat_prc-process_id
      process_step_no   = is_mat_prc-process_step_no
      source_system     = is_mat_prc-source_system
      source_id         = is_mat_prc-source_id        BINARY SEARCH.

 IF sy-subrc IS INITIAL.
    MOVE-CORRESPONDING <prc> TO ls_zkmi.
    ls_zkmi-matnr = iv_matnr.
    APPEND ls_zkmi TO cs_mat_data-zkmi_tab.
  ENDIF.

ENDMETHOD.
```

CALL_API_EXTENSION_PREPARE

```
METHOD call_api_extension_prepare.

  DATA ls_zkmi     TYPE zkmi.
  DATA ls_zkmi_x TYPE zkmi_x.
  DATA ls_mat_data TYPE cmd_bs_mat_s_mara.
  DATA ls_mat_segments_ext LIKE LINE OF et_mat_segments_ext.

  CLEAR: et_mat_segments_ext.

  READ TABLE is_mat_data-mara_tab INTO ls_mat_data INDEX 1.

  ls_mat_segments_ext = 'ZKMI_TAB'.
  INSERT ls_mat_segments_ext INTO TABLE et_mat_segments_ext.

  LOOP AT is_mat_data-zkmi_tab INTO DATA(ls_zkmitab) WHERE matnr EQ ls_mat_data-matnr.
    MOVE-CORRESPONDING ls_zkmitab TO ls_zkmi.
    ls_zkmi-matnr = iv_matnr.
    INSERT ls_zkmi INTO TABLE cs_mat_data-zkmi_tab.
  ENDLOOP.

  LOOP AT is_mat_data-zkmi_x_tab INTO DATA(ls_zkmitab_x) WHERE matnr EQ ls_mat_data-matnr.
    MOVE-CORRESPONDING ls_zkmitab_x TO ls_zkmi_x.
    ls_zkmi_x-matnr  = iv_matnr.
    ls_zkmi_x-zaccp  = 'X'.
    ls_zkmi_x-zfltp  = 'X'.
    ls_zkmi_x-ztims  = 'X'.
    ls_zkmi_x-zxfeld = 'X'.
    INSERT ls_zkmi_x INTO TABLE cs_mat_data-zkmi_x_tab.
  ENDLOOP.

ENDMETHOD.
```