PUBLIC

How-To: Node (Table) Extension for Material/Product in MDG Consolidation and Mass Processing (Releases up to S/4HANA 1909)

Applicable Releases:

SAP MDG, Consolidation and SAP MDG, Mass Processing running on release SAP MDG 9.0 or higher and from SAP S/4HANA 1610 until S/4HANA 1909**.**

As of S/4HANA 2020 there was a change in the API for Material/Product. There is a separate how-to guide for releases from S/4HANA 2020 on.

Version 1.4

April 2024

Document History

Document Version	Description
1.0	First official release of this guide
1.1 -1.3	Minor content updates
1.4	New format (template)

1 Introduction

This document explains how to add a new node to the Material data model.

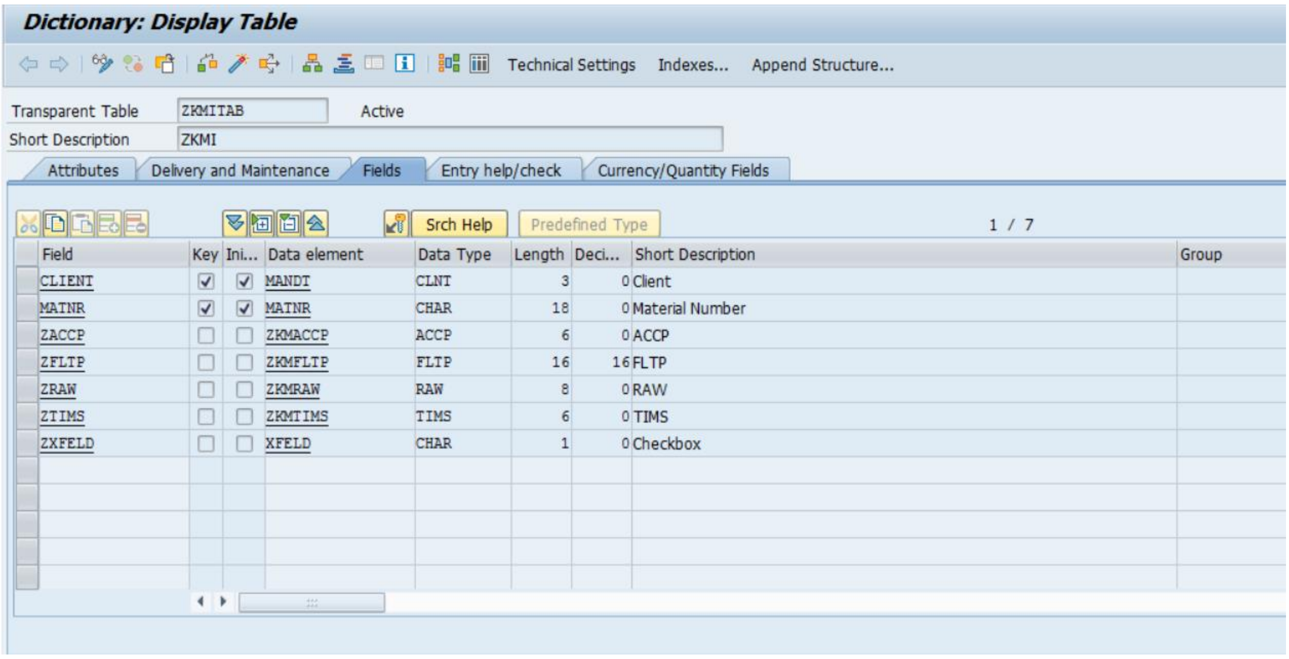
We recommend studying the following how-to guides before working with this one that describes the back-end extension:
Extend MDG Material – Node Extension (Reuse Option) for standard ERP tables
Extend MDG Material – Node Extension (Reuse Option) for custom tables.

2 Business Scenario

During a consolidation or mass process, the additional data will be loaded into MDG consolidation source tables and/or process tables and be processed along with standard data. After activation, that data will then be stored in a custom Z-table.

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

Remark: MDG Flex tables are not supported! Only tables in customer namespace are supported.



Transparent Table ZKMITAB Active
Short Description ZKMI

Field	Key	Ini...	Data element	Data Type	Length	Deci...	Short Description	Group
CLIENT	☑	☑	MANDT	CLNT	3	0	Client	
MATNR	☑	☑	MATNR	CHAR	18	0	Material Number	
ZACCP	□	□	ZKMACCP	ACCP	6	0	ACCP	
ZFLTP	□	□	ZKMFLTP	FLTP	16	16	FLTP	
ZRAW	□	□	ZKMRAW	RAW	8	0	RAW	
ZTIMS	□	□	ZKMTIMS	TIMS	6	0	TIMS	
ZXFELD	□	□	XFELD	CHAR	1	0	Checkbox	

Extended ERP Material Data

3 Step by Step Implementation Guide

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 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 structure CMD_BS_MAT_S_MAT_DATA
   b. To implement the extension for SAP BAdI CMD_BS_MAT_API_SEGMENTS_EXT to check and save customer table data.

      Important: When running MDG on S/4HANA 2020 or higher, please consider the new how-to guide!

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 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 new class ZCL_MDC_DATA_MAT that inherits from class CL_MDC_DATA_MAT. The new class will then be used in consolidation processes involving Process Model 194 (Material Data). Note that for all customer extensions of 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 suffixes _SRC and _PRC. This method can also be redefined to use 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

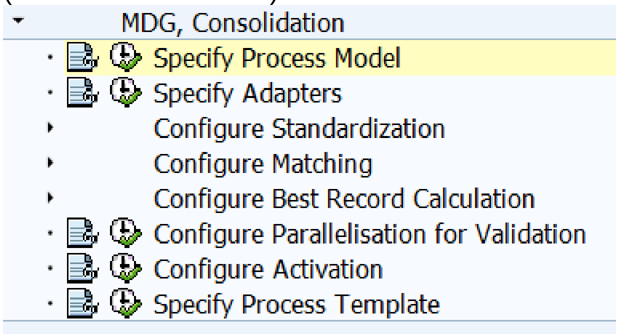
If your custom field uses a “Large Object Binary” related data type (e.g. a string, blob, raw binary or similar), you need to redefine one more method in your custom Product Data Access class, namely:

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

3.2 Extend Process Model

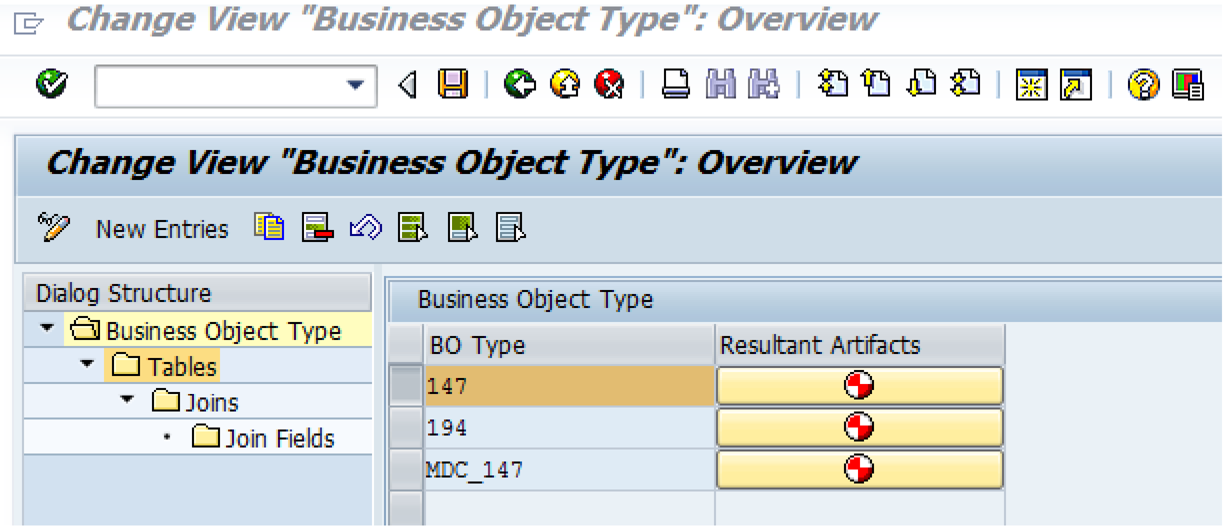
To extend the Process Model with a new table, the contents of View Cluster VC_MDC_MODEL must be changed. This view cluster contains the process model, which includes all relevant tables of an object and their relations.

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



MDG, Consolidation
- Specify Process Model
- Specify Adapters
- Configure Standardization
- Configure Matching
- Configure Best Record Calculation
- Configure Parallelisation for Validation
- Configure Activation
- Specify Process Template

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



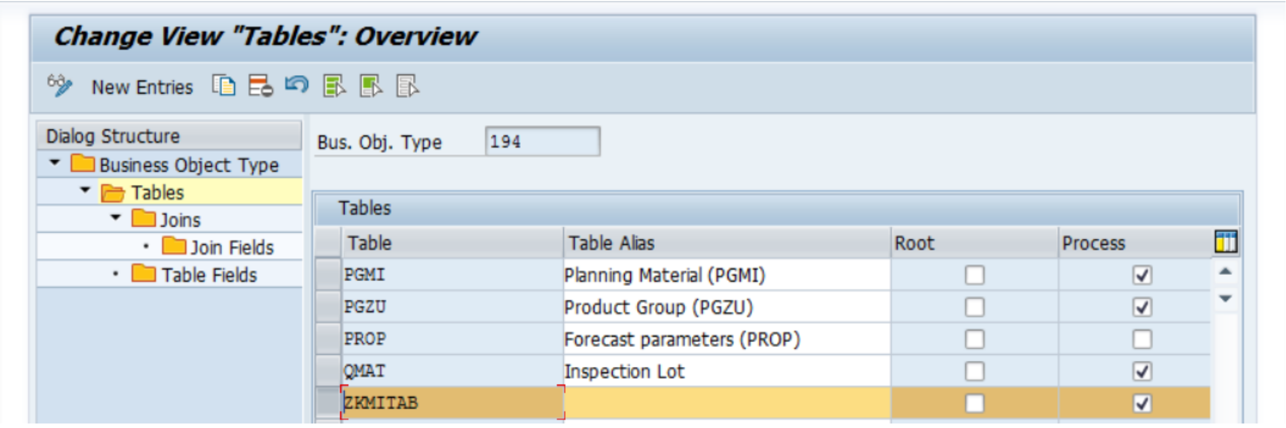
Change View "Business Object Type": Overview

Dialog Structure
- Business Object Type
  - Tables
    - Joins
      - Join Fields

Business Object Type

BO Type	Resultant Artifacts
147	●
194	●
MDC_147	●

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.



Change View "Tables": Overview

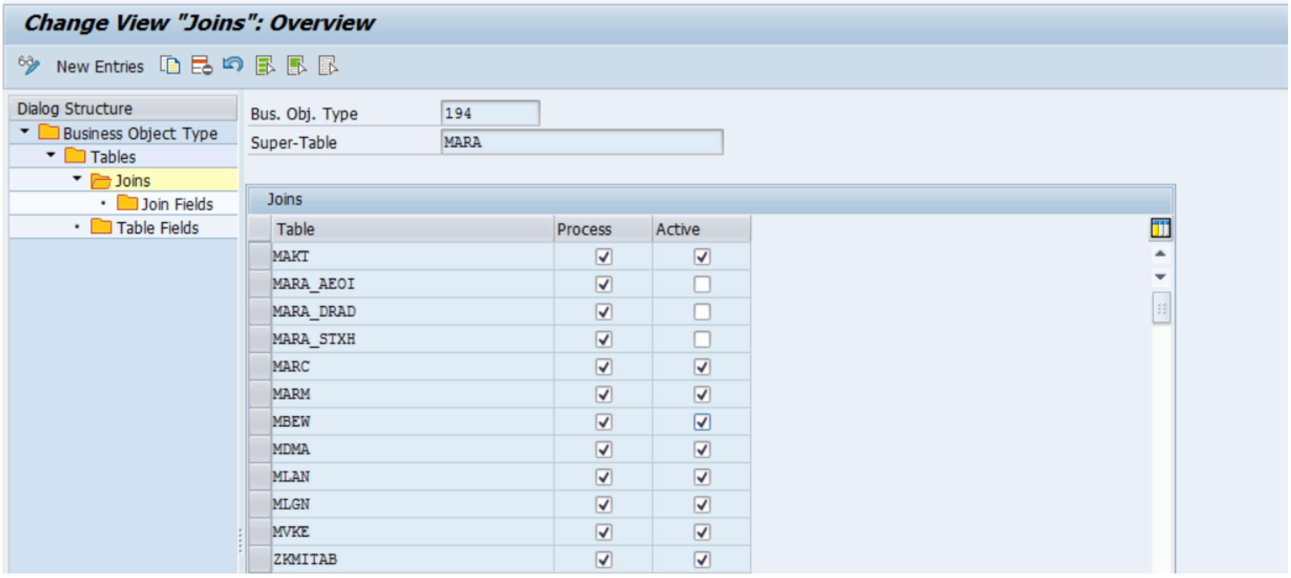
Dialog Structure
- Business Object Type
  - Tables
    - Joins
      - Join Fields
    - Table Fields

Bus. Obj. Type: 194

Tables

Table	Table Alias	Root	Process
PGMI	Planning Material (PGMI)	□	☑
PGZU	Product Group (PGZU)	□	☑
PROP	Forecast parameters (PROP)	□	□
QMAT	Inspection Lot	□	☑
ZKMITAB		□	☑

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



Change View "Joins": Overview

Dialog Structure
- Business Object Type
  - Tables
    - Joins
      - Join Fields
    - Table Fields

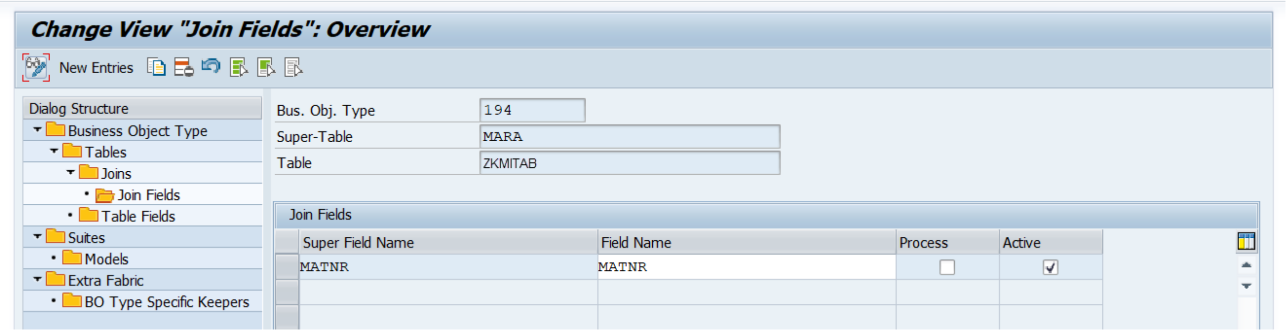
Bus. Obj. Type: 194

Super-Table: MARA

Joins

Table	Process	Active
MAKT	☑	☑
MARA_AEOI	☑	□
MARA_DRAD	☑	□
MARA_STXH	☑	□
MARC	☑	☑
MARM	☑	☑
MBEW	☑	☑
MDMA	☑	☑
MLAN	☑	☑
MLGN	☑	☑
MVKE	☑	☑
ZKMITAB	☑	☑

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



Change View "Join Fields": Overview

Dialog Structure
- Business Object Type
  - Tables
    - Joins
      - Join Fields
    - Table Fields
- Suites
  - Models
- Extra Fabric
  - BO Type Specific Keepers

Bus. Obj. Type: 194

Super-Table: MARA

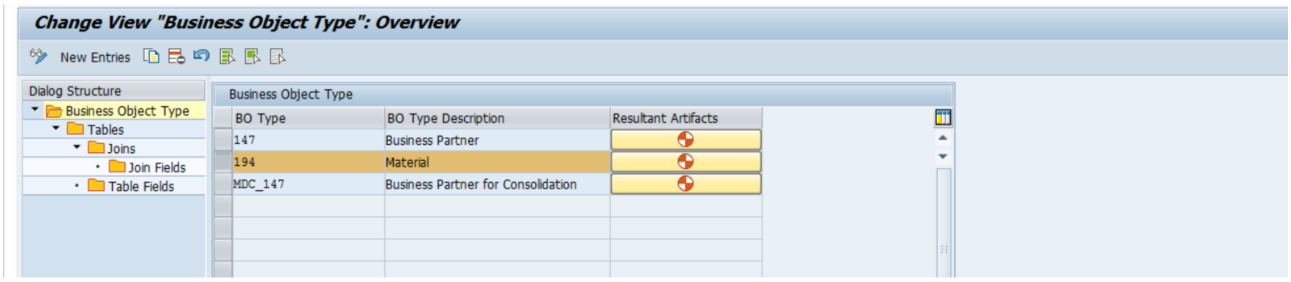
Table: ZKMITAB

Join Fields

Super Field Name	Field Name	Process	Active
MATNR	MATNR	□	☑

7. Save the changes.

8. Generate artifacts for new table ZKMITAB for Business Object Type 194.



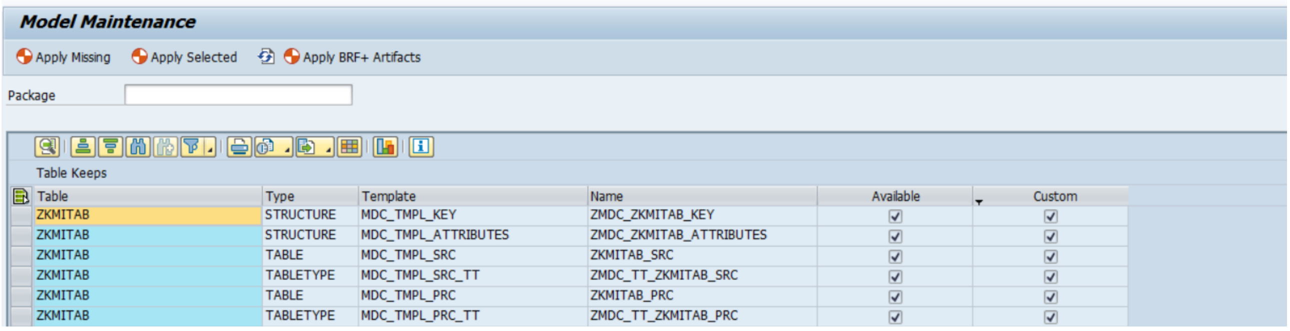
Change View "Business Object Type": Overview

Dialog Structure
- Business Object Type
  - Tables
    - Joins
      - Join Fields
    - Table Fields

Business Object Type

BO Type	BO Type Description	Resultant Artifacts
147	Business Partner	
194	Material	
MDC_147	Business Partner for Consolidation	

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.)



Package

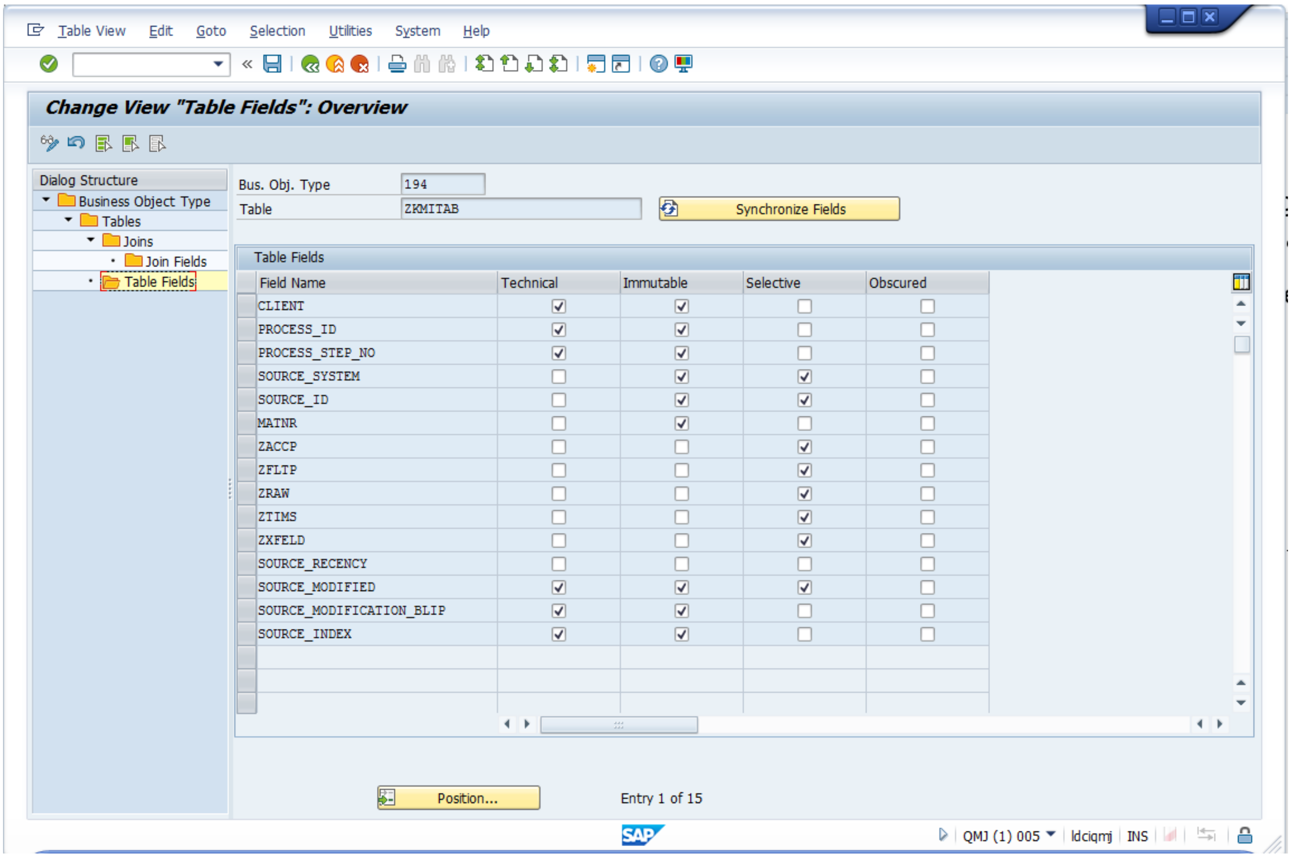
Table Keeps

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.



Bus. Obj. Type 194

Table ZKMITAB

Table Fields

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	☑	☑	□	□

Entry 1 of 15

3.3 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 class CL_MDC_MODEL_MAT. The new class will then be used in processes involving Process Model 194 (Material Data). Note that for all material extensions of 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

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

3.4 Extend MDG for writing new table into Change Request

If all MDG extensions are done as described, the data is written generically into a Change Request using SMT (field mapping extension).

Result:

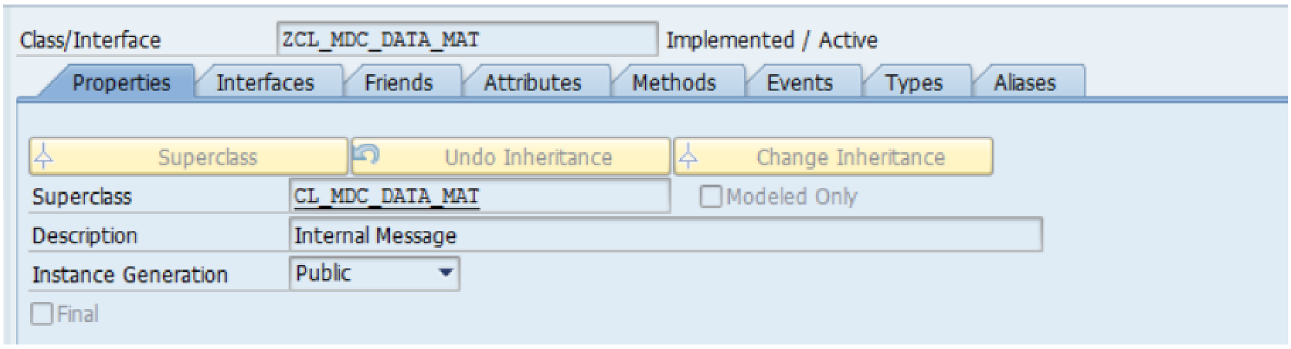
The process model for Material Data has been extended by a new table. The processed result can be written into an active database table and into a Change Request.

4 Appendix

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.



Class/Interface: ZCL_MDC_DATA_MAT Implemented / Active

Superclass: CL_MDC_DATA_MAT □ Modeled Only

Description: Internal Message

Instance Generation: Public

□ Final

Method Redefinitions

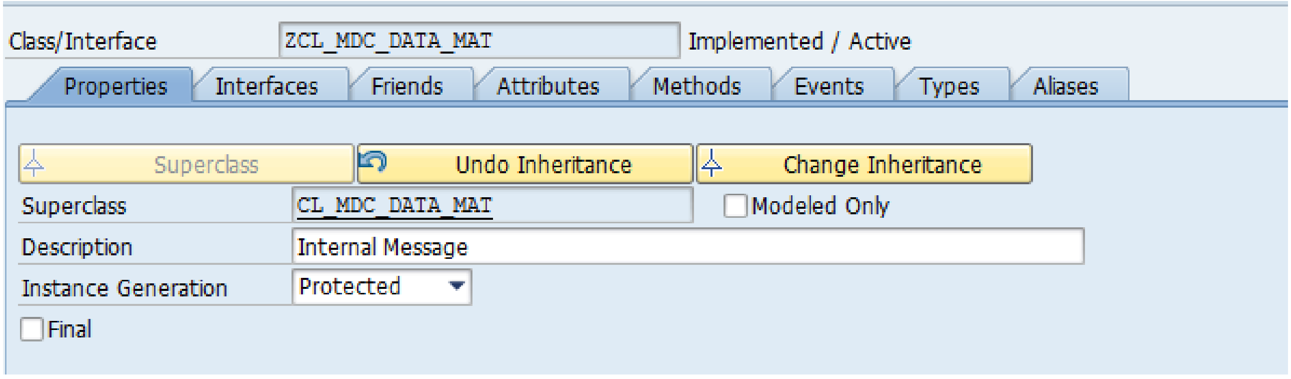
In the given example for table ZKMITAB no methods need to be redefined, so the table natively maps via MATNR (and other attributes) to active data.

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.



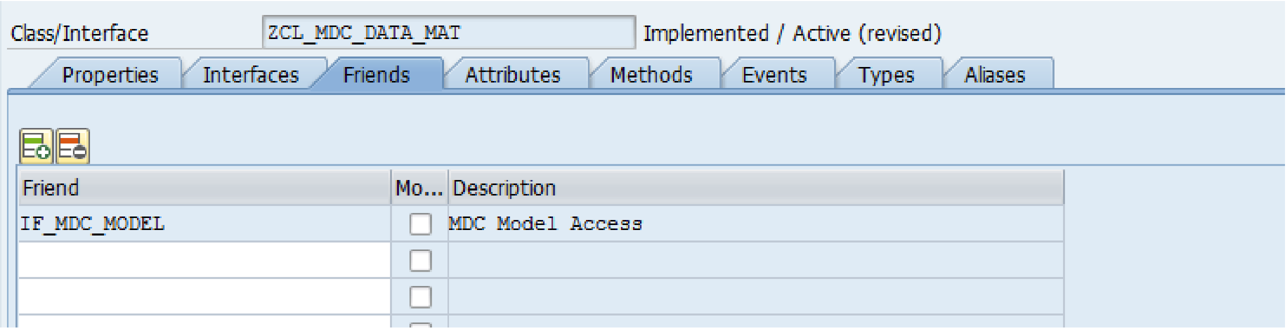
Class/Interface: ZCL_MDC_DATA_MAT Implemented / Active

Superclass: CL_MDC_DATA_MAT □ Modeled Only

Description: Internal Message

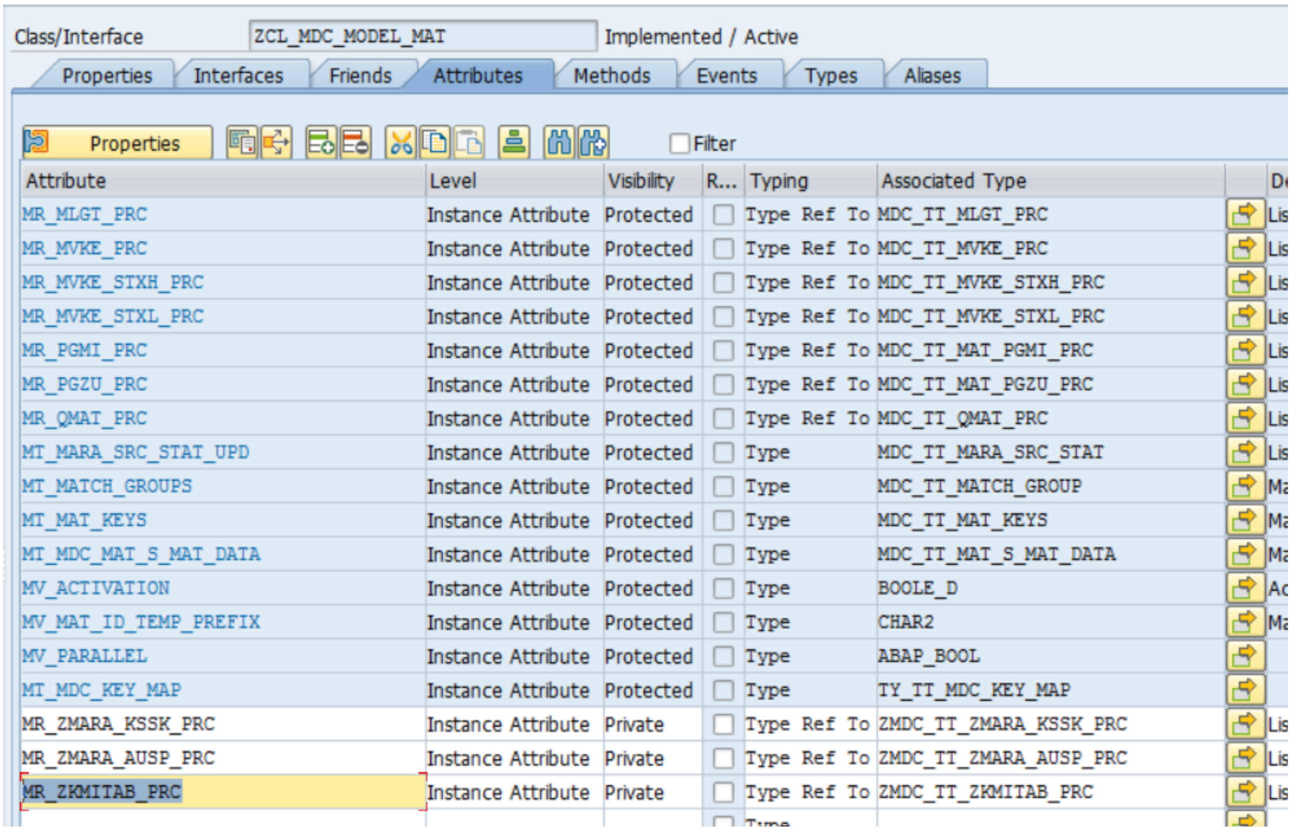
Instance Generation: Protected

□ Final



Class/Interface: ZCL_MDC_DATA_MAT Implemented / Active (revised)

Friend	Mo...	Description
IF_MDC_MODEL	□	MDC Model Access
	□	
	□	



Class/Interface ZCL_MDC_MODEL_MAT Implemented / Active

Attribute	Level	Visibility	R...	Typing	Associated Type		De
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.
```