SAP How-To Guide for MDG-F

Cross-entity Derivation

Applies to

Master Data Governance for Financials (MDG-F) with release version 7.0 and newer. For more information, visit the Master Data Management homepage (https://go.sap.com/community/topic/master-data-governance.html).

Summary

SAP Master Data Governance provides out-of-the box solutions for the central management of master data objects. Domain-specific solutions include business partner (MDG-BP), customer (MDG-C), supplier (MDG-S) governance, material governance (MDG-M), and financials governance (MDG-F).

If your domain-specific solution does not fully meet requirements, you can customize and extend it.

This guide provides you with the foundation knowledge you need to extend the MDG-F data model by new fields.

Author(s): Michael Theis

Company: SAP SE

Created on: October 2016

Version: 1.2

Table of Contents

Introduction

SAP Master Data Governance (MDG) is used for embedded Master Data Management (MDM), that is, centralized, out-of-the-box, domain-specific creation, modification, and distribution of master data with a focus on SAP Business Suite.

Domain-specific content (data models, user interfaces, workflows) is provided as part of the standard for several application areas. It is a common requirement from customers to adapt the MDG data models to their specific needs.

This document explains how to implement a custom cross-entity derivation for MDG-F entity types. It covers the key concepts and implementation details in general and includes a real-life example of the MDG-F data model **0G**.

We recommend that you study the following how-to guide before working with the current guide:

Extensibility Options for SAP Master Data Governance for Financial Data

- Overview about MDG-F

MDG for Financials

MDG offers a domain-specific solution for financial governance (MDG-F). The current MDG-F data model is called **0G**. It covers entity types of the accounting, controlling and consolidation components of financial master data as indicated by the examples below:

- Accounting: ***G/L Account*** (**ACCOUNT & ACCCCDET**), ***Company***
- Controlling: ***Cost Center*** (**CCTR**), ***Cost Element*** (**CELEM**) and ***Profit Center*** (**PCTR**)
- Consolidation: ***Consolidation Unit*** **(CONSUNIT)** , ***Item*** **(FSI)**

Scenario

The MDG-F data model **0G** includes the attribute **Person Responsible** for Cost Centers. This attribute has to be filled in by the end-user since it is usually mandatory. Using a data derivation it is possible to automate this, for example during the creation of a new cost center.

Prerequisites

Mandatory: You have applied SAP Note 2006227 in your system.

Technical Background

Having read the overview document Extensibility Options for SAP Master Data Governance for Financial Data ➔ Overview about MDG-F you already know that the data derivation in MDG-F is implemented using the rule service BAdI **`USMD_RULE_SERVICE_CROSS_ET`**. Therefore the implementation requires some knowledge in this area as well as ABAP coding skills.

Cross-Entity Derivation

Cross-entity derivation offers the best flexibility for data derivation. The BAdI is called with an external instance of the USMD model and the data that has been changed. The changed data contains additional meta-information about the actual change (for example, an indication of whether it is new data or deleted data, and in case of updated data, a list of changed attributes). This enables the implementation of very specific data derivations. Additionally, it is possible to change the data of related entity types (for example, based on a creation of a new entity with ***SU Type*** 1, some dependent records of a child entity with ***SU Type*** 4 are created automatically).

Cross-entity derivation is executed for both data maintenance from the single-object user interfaces and the data import using the Data Import Framework. It is important to keep this in mind for the implementation of a custom data derivation.

The disadvantage of cross-entity derivation is that it is called for each entity type of data model **`0G`**. So, the first task of a custom data derivation is to determine what has exactly changed and if a data derivation is needed at all. This can add some complexity to the custom coding. Furthermore, it is not possible to derive data across different entities with ***SU Type*** 1.

Predefined SAP Implementation for MDG-F

MDG-F uses the cross-entity derivation for some of its entity types. It is strongly recommended to make yourself familiar with the existing coding. This could simplify your custom implementation since some of the required principles are already implemented by SAP.

Abstract Implementation Class

Check class **`CL_USMDZ7_RS_CROSS_ENTITY`**. This class is the main entry point for all MDG-F related derivations. Method **`IF_EX_USMD_RULE_SERVICE2~DERIVE`** shows how to identify the current entity type that has been changed. Furthermore, it delegates the entity-specific derivations to separate classes. This could be an optional step for your custom implementation, too.

Derivations for Accounts in Company Code

The default data derivation for Cost Centers is implemented in class **`CL_USMDZ7_RS_ACCCCDET`** in method **`IF_EX_USMD_RULE_SERVICE2~DERIVE`**. The implementation provides data derivation for both the creation of new accounts in the company code as well as the change of existing accounts in the company code:

New Accounts in Company Code trigger the following derivations:

- New Company Code → Derive Currency
  Runs always to ensure data consistency. The user input respectively data import has priority but might be adjusted according to the company code customizing.

Changed Accounts in Company Code trigger the following derivations:

- Validation for key and attribute changes at the same time.
  The validation is needed to ensure data consistency when both the object keys and certain attributes are maintained at the same point in time. This results in calling the derivation twice (first only keys, second with changed data). Affected attributes are: Currency.

Derivations for Cost Centers

The default data derivation for Cost Centers is implemented in class **`CL_USMDZ7_RS_CCTR`** in method **`IF_EX_USMD_RULE_SERVICE2~DERIVE`**. The implementation provides data derivation for both the creation of new cost centers as well as the change of existing cost centers:

New Cost Centers trigger the following derivations:

1. New Controlling Area → Derive Company Code
   Runs only if the company code is not set in the current round-trip. The user input respectively data import has priority since consistency checks are ensured by the framework.
2. New Controlling Area → Derive Language
   Runs only if the language is not yet set.
3. New Controlling Area → Derive Logical System
   Runs always to ensure data consistency. In the UI the logical system is always read-only. In data import the value might be aligned with the sending system only.
4. New/derived Company Code → Derive Currency
   Runs always to ensure data consistency. The user input respectively data import has priority but might be adjusted according to the controlling area vs. company code customizing.
5. New/derived Company Code → Derive Functional Area
   Runs always to ensure data consistency. If the current company code does not use functional areas, its value must be empty.
6. New Cost Center Category → Derive Indicators and Functional Area
   Runs only for the creation of new cost centers. User input is overwritten but import data has priority. Functional area is only derived if the current company code allows the same.

Changed Cost Centers trigger the following derivations:

1. Validation for key and attribute changes at the same time

The validation is needed to ensure data consistency when both the object keys and certain attributes are maintained at the same point in time. This results in calling the derivation twice (first only keys, second with changed data). Affected attributes are: Company Code, Currency, and Functional Area

2. Changed Company Code → Derive Currency (see above)
3. Changed Company Code → Derive Functional Area (see above)
   Changed Cost Center Category → Derive Indicators and Functional Area (see above)

Derivations for Cost Elements

The default data derivation for Cost Elements is implemented in class **`CL_USMDZ7_RS_CELEM`** in method **`IF_EX_USMD_RULE_SERVICE2~DERIVE`**. The implementation provides data derivation for both the creation of new cost elements as well as the change of existing cost elements:

New Cost Elements trigger the following derivations:

1. New Cost Element ID → Name (Short Text)
   Runs only for PRIMARY cost elements and if the Name is not set in the current round-trip. The user input or the data import has priority since consistency checks are ensured by the framework.
2. New Cost Element ID → Medium Text
   Runs only for PRIMARY cost elements and if the Medium Text is not set in the current round-trip. The user input or the data import has priority since consistency checks are ensured by the framework. The medium text is derived from long text of the account.
3. New Cost Element ID → Functional Area
   Runs only for PRIMARY cost elements. It is mandatory to reuse the value of the related account.
4. New Cost Element ID → Cost Element Category
   Runs for PRIMARY cost elements only and if the category is not set in the current round-trip. The user input respectively data import has priority since consistency checks are ensured by the framework.

Changed Cost Elements trigger the following derivations:

1. Changed Functional Area → derive Functional Area
   Runs for cost elements only to ensure that the functional area is the one of the account.

Derivations for Profit Centers

The default data derivation for Profit Centers is implemented in class **`CL_USMDZ7_RS_PCTR`** in method **`IF_EX_USMD_RULE_SERVICE2~DERIVE`**. The implementation provides data derivation for both the creation of new profit centers as well as the change of existing profit centers:

New and changed Profit Centers both trigger the following derivations:

1. New Profit Center → assign all valid Company Codes
   The standard behavior for the creation of a new profit center is the assignment of all valid company

codes. This should be done automatically for SOM UI and File Upload. File Import might already provide the assignments.

2. New or changed Company Code Assignment → derive Profit Center

   The profit center contains the indicator 'Postable in All Company Codes' (PCTR-PCTRCCALL). This field is hidden in the SOM UI. It is set if all valid Company Codes are assigned to the profit center, respectively removed it this is not the case.

Implementation

The implementation consists of several steps. The list below relates to the planned scenario of the current how-to guide. It may differ for your use case.

1. Prepare the custom data derivation by creating a custom class.

   You do this only once, even if you decide to add more data derivations later.

2. Create a custom Enhancement Implementation for **`USMD_RULE_SERVICE_CROSS_ET`**.

   You do this only once, even if you decide to add more data derivations later.

3. Test your custom Enhancement Implementation.

   Testing ensures that both your custom enhancement implementation and the SAP-defined data derivations are working.

4. Implement your custom Data Derivation.
   a. This step explains how to derive the persons responsible of a cost center while the cost center is created. Use the example code for your own derivations.
   b. You may repeat this step for all derivations you want to create – even in later points in time.

Step 1: Create Your Custom Class for the BAdI

The scenario creates as new implementation for the BAdI **USMD_RULE_SERIVE_CROSS_ET**. This requires an implementation class for the BAdI which should be prepared before creating the enhancement.

1. Log on to your MDG hub system.
2. Start transaction **SE24**.
3. Create a new class that inherits from the SAP standard class **CL_USMDZ7_RS_CROSS_ENTITY**, for example, **ZCL_MDGF_RS_CROSS_ET**
   It is mandatory that your custom class inherits from the SAP standard class **CL_USMDZ7_RS_CROSS_ENTITY**. Otherwise, the default data derivation provided by SAP cannot be executed.
4. Create a re-definition of method **IF_EX_USMD_RULE_SERVICE2~DERIVE** that consists only of the parent call. The parent call is mandatory. Otherwise, the default data derivation provided by SAP cannot be executed.

```
"Inherit first to ensure that all SAP derivations are executed correctly.
super->if_ex_usmd_rule_service2~derive(
  EXPORTING
    io_model        = io_model
    io_changed_data = io_changed_data
    io_write_data   = io_write_data
  IMPORTING
    et_message_info = et_message_info ).
```

5. Save and activate the class.

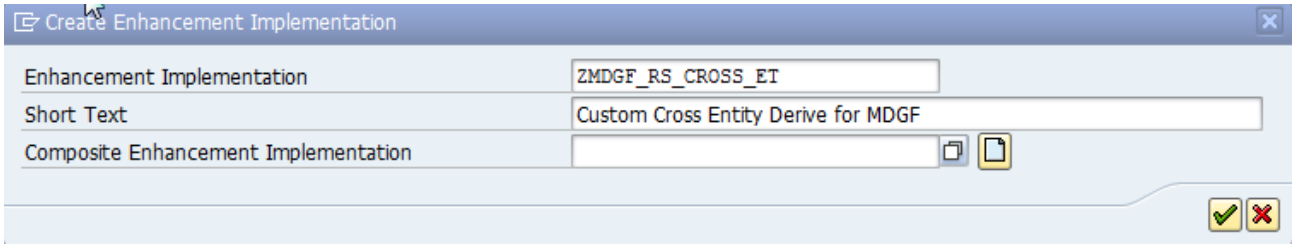
Result

You have prepared the implementation class for the BAdI that you defined in the next step.

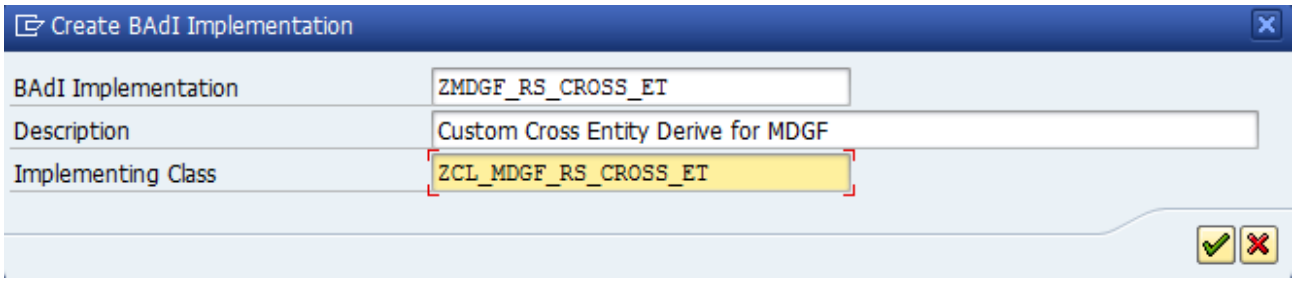
Step 2: Create Your Custom Enhancement.

The scenario creates a new implementation for the BAdI **`USMD_RULE_SERIVE_CROSS_ET`**. You use the implementation class created by the previous step.

1. Log on to your MDG hub system.
2. Start transaction **`SE18`**.
3. Choose enhancement spot **`USMD_RULE_SERVICE`**.
4. Select BAdI Definition **`USMD_RULE_SERVICE_CROSS_ET`**.
5. Right-click the selected entry and choose *Create BAdI Implementation* from the context menu.
6. In pop-up *Select or Create Enhancement Implementation* choose the **New** button to create a new enhancement implementation first.

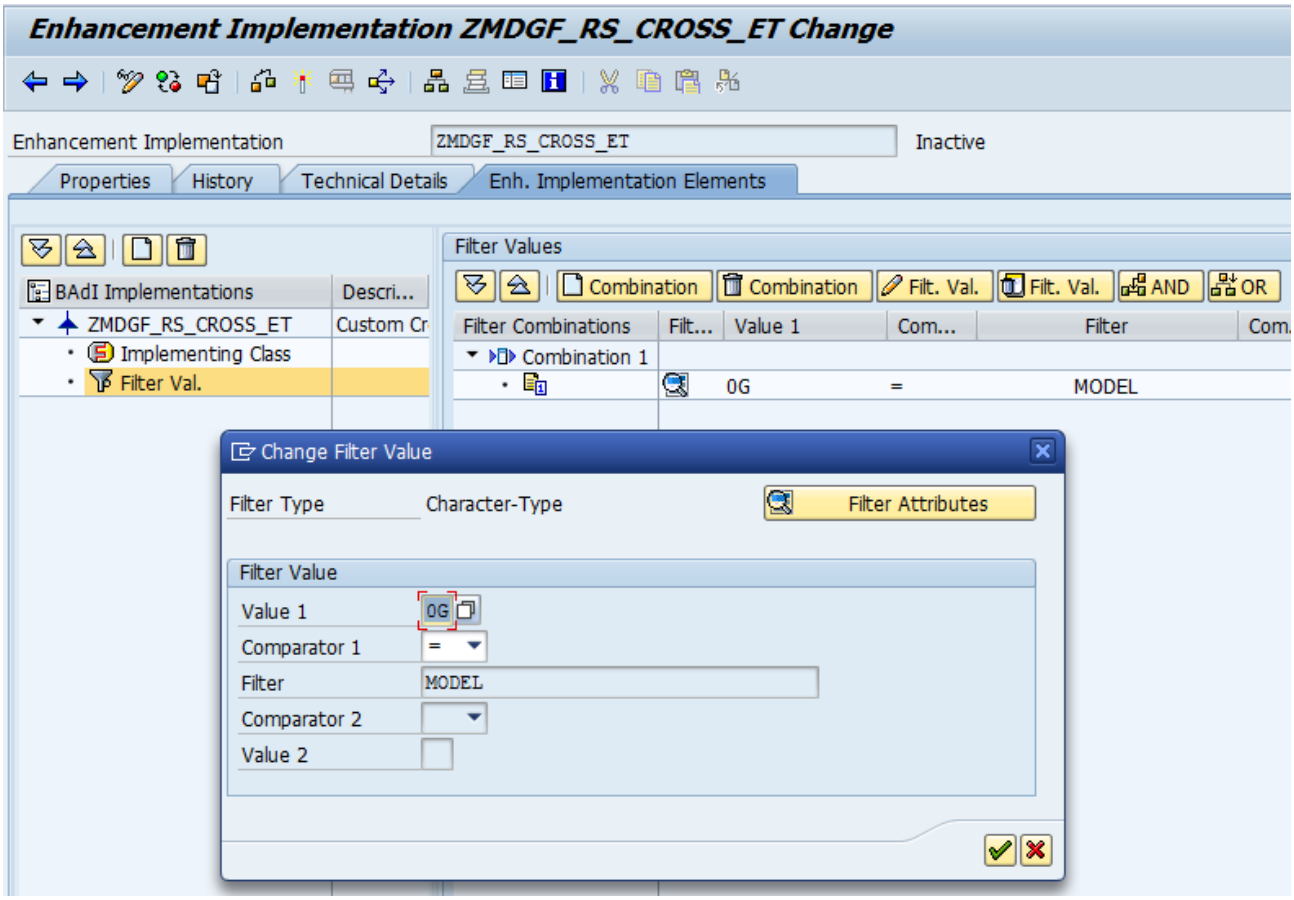


7. In the pop-up, ***Select or Create Enhancement Implementation*** double-click on your newly created enhancement implementation.
8. In pop-up ***Create BAdI Implementation*** define a new BAdI using the BAdI Implementation Class that you have created in step 1.



9. If you receive a selection pop-up asking if you want to re-use an example implementation, **cancel** this pop-up. Make sure you use your custom class.

10. Back in the enhancement details view, create a new **Filter Value** for your enhancement implementation. It is mandatory to create a filter for data model **0G**.



11. Save and activate the Enhancement Implementation including the BAdI.

Result

You have successfully created a new enhancement using your custom BAdI implementation class. Proceed with testing the same.

Step 3: Test the Custom Enhancement

The test of the custom enhancement calls the following classes:

- The custom implementation class
  Calling this class ensures your custom data derivations are executed.
- The predefined SAP implementation
  Calling this class ensure that the default derivations are still executed.

1. Logon to your MDG hub system.
2. Set breakpoints in method **`IF_EX_USMD_RULE_SERVICE2~DERIVE`** in both classes **`CL_USMDZ7_RS_CROSS_ENTITY`** (the default SAP implementation) and **`ZCL_MDGF_RS_CROSS_ET`** (your custom class).
3. Start the single object maintenance user interface for cost centers.
4. Create a new cost center.
5. Maintain the key fields and trigger a UI round-trip for each key field. For example for when the user presses **ENTER** or chooses the ***Check*** button after maintaining the ***Cost Center ID***.
6. The data derivation is executed during the round-trip, but only if data has been maintained.
7. Check that the breakpoint in your custom class is reached.

Result

The custom enhancements as well as the SAP default implementation are both working. You can implement your custom data derivation.

Step 4: Implement Your Custom Derivation

The scenario sets the value of the "Person Responsible" attribute for all new Cost Center(s) to "Mr. X" if no specific person has been maintained by the user (respectively by data import).

1. Log on to your MDG hub system.
2. Go to method **`IF_EX_USMD_RULE_SERVICE2~DERIVE`** in class **`ZCL_MDGF_RS_CROSS_ET`** (your custom class).
3. Switch to change mode and copy and paste the complete source code as given in the appendix.
4. The general logic of a derive implementation consists of the steps:
    a. Call the parent implementation to ensure that the SAP default derivations are executed.
    b. Analyze the changed entities to decide whether a custom derivation is required or not.
        i. Method **`IO_CHANGED_DATA_->GET_ENTITY_TYPES`** supports various exporting parameters. It is possible to ask for deleted entity types only.
        ii. Method **`IO_CHANGED_DATA->READ_DATA`** can return new, changed, or deleted **data** for the requested entity type. Use the exporting parameter according to your custom derivation. The example code uses a very simple approach. More complex handling, including the further processing, is implemented by the SAP default derivation.
    c. Prepare the references needed for writing the derived data to **`IO_WRITE_DATA`**.
        iii. Use **`IO_WRITE_DATA->CREATE_DATA_REFERENCE`** to ensure that you work with the correct data structures. Otherwise, the system might cause a short dump.
        iv. The current edition is always mandatory for writing derived data. If you do not determine the edition, the system causes a short dump.
    d. Execute the derivations.
    e. Write the derived data back to **`IO_WRITE_DATA`**.
        v. Ensure that **`IO_WRITE_DATA->WRITE_DATA`** is always called with the current object keys and the edition in **`IT_KEYS`** as well as the changed attribute(s) listed in **`IT_ATTRIBUTE`**. This ensures that you do not overwrite any SAP default derivation by accident.
5. Save and activate the changes.

Result

The data derivation is successfully implemented. Test it by creating a new cost center using the single object maintenance user interfaces.

Additional Information

Links

FPM on SCN

MDG Guides on Service Market Place

How-to Guides

Extensibility Options for SAP Master Data Governance for Financial Data

- Overview about MDG-F

Version History

- 1.2 – Updated new SAP Community Links
- 1.1 – Updated SAP pre-defined Derivations
- 1.0 – First release of the document

Appendix – Source Code of ZCL_MDGF_CROSS_ET

Method IF_EX_USMD_RULE_SERVICE2~DERIVE

```
METHOD if_ex_usmd_rule_service2~derive.
*! The example derivation sets the value of the "Person Responsible"
*  attribute for all new Cost Center(s) to "Mr. X" if no specific person
*  has been maintained by the user (respectively by data import).
  CONSTANTS:
    gc_person TYPE usmd_fieldname VALUE 'CCTRRESPP'.

  DATA:
    lo_context TYPE REF TO if_usmd_app_context,
    lr_derived_data_struc TYPE REF TO data,
    lr_derived_data_table TYPE REF TO data,
    lr_new_data           TYPE REF TO data,
    ls_key                TYPE usmd_s_value,
    lt_attribute          TYPE usmd_ts_fieldname,
    lt_key                TYPE usmd_ts_value,
    lt_changed_entities   TYPE usmd_t_entity,
    lv_edition            TYPE usmd_edition.

  FIELD-SYMBOLS:
    <ls_derived_data> TYPE any,
    <ls_new_data>     TYPE any,
    <lt_derived_data> TYPE ANY TABLE,
    <lt_new_data>     TYPE ANY TABLE,
    <lv_value>        TYPE any.

  "Inherit first to ensure that all SAP derivations are executed correctly.
  super->if_ex_usmd_rule_service2~derive(
    EXPORTING
      io_model        = io_model
      io_changed_data = io_changed_data
      io_write_data   = io_write_data
    IMPORTING
      et_message_info = et_message_info ).

  "Get changed entities.
  io_changed_data->get_entity_types(
    IMPORTING
      et_entity     = lt_changed_entities ).

  "Check if cost centers have been changed.
  READ TABLE lt_changed_entities TRANSPORTING NO FIELDS
    WITH KEY table_line = if_usmdz_cons_entitytypes=>gc_entity_cctr.
  IF sy-subrc NE 0.
    "No cost center in changed data.
    RETURN.
  ENDIF.
```

```
"Check if there are NEW cost centers.
io_changed_data->read_data(
  EXPORTING
    i_entity               = if_usmdz_cons_entitytypes=>gc_entity_cctr
  IMPORTING
    er_t_data_ins          = lr_new_data ).
IF lr_new_data IS NOT BOUND.
  "No new cost centers.
  RETURN.
ENDIF.
ASSIGN lr_new_data->* TO <lt_new_data>.
IF sy-subrc NE 0
  OR <lt_new_data> IS INITIAL.
  "Final check to prevent a short dump.
  RETURN.
ENDIF.

"There are new cost centers. The derivation has to be done using the
"IO_WRITE_DATA parameter. This needs some preparations.
TRY.
    lr_derived_data_table = io_write_data->create_data_reference(
                              i_entity     = if_usmdz_cons_entitytypes=>gc_entity_cctr
                              i_struct     = io_model->gc_struct_key_attr ).
    ASSIGN lr_derived_data_table->* TO <lt_derived_data>.
    CREATE DATA lr_derived_data_struc LIKE LINE OF <lt_derived_data>.
    ASSIGN lr_derived_data_struc->* TO <ls_derived_data>.
  CATCH cx_usmd_write_error.
    "You could transform the exception to ET_MESSAGE_INFO if desired.
ENDTRY.

"The current edition must be part of the data handed over to IO_WRITE_DATA.
TRY.
    lo_context ?= cl_usmd_app_context=>get_context( ).
  CATCH cx_sy_move_cast_error.
    RETURN.
ENDTRY.
IF lo_context IS NOT BOUND.
  RETURN.
ENDIF.
lo_context->get_attributes( IMPORTING ev_edition = lv_edition ).
IF lv_edition IS INITIAL.
  RETURN.
ENDIF.

"Handle the new cost center(s)
LOOP AT <lt_new_data> ASSIGNING <ls_new_data>.
  CLEAR: <ls_derived_data>, <lt_derived_data>,
         ls_key, lt_attribute, lt_key.
```

```
    "Check the current value of the responsible person. Given values
    "should not be overwritten.
    ASSIGN COMPONENT gc_person OF STRUCTURE <ls_new_data> TO <lv_value>.
    CHECK sy-subrc EQ 0
      AND <lv_value> IS INITIAL.

    "Prepare the object keys for the derive.
    ASSIGN COMPONENT if_usmdz_cons_entitytypes=>gc_entity_coarea
      OF STRUCTURE <ls_new_data> TO <lv_value>.
    CHECK sy-subrc EQ 0
      AND <lv_value> IS NOT INITIAL.
    ls_key-fieldname = if_usmdz_cons_entitytypes=>gc_entity_coarea.
    ls_key-value = <lv_value>.
    INSERT ls_key INTO TABLE lt_key.
    ASSIGN COMPONENT if_usmdz_cons_entitytypes=>gc_entity_cctr
      OF STRUCTURE <ls_new_data> TO <lv_value>.
    CHECK sy-subrc EQ 0
      AND <lv_value> IS NOT INITIAL.
    ls_key-fieldname = if_usmdz_cons_entitytypes=>gc_entity_cctr.
    ls_key-value = <lv_value>.
    INSERT ls_key INTO TABLE lt_key.
    ls_key-fieldname = usmd0_cs_fld-edition.
    ls_key-value = lv_edition.
    INSERT ls_key INTO TABLE lt_key.

    "Derive the person responsible.
    ASSIGN COMPONENT gc_person OF STRUCTURE <ls_derived_data> TO <lv_value>.
    CHECK sy-subrc EQ 0.
    <lv_value> = 'Mr. X'.

    "Indicate that the changed attributes. This is important to prevent
    "overwriting SAP derivations!
    INSERT gc_person INTO TABLE lt_attribute.

    "Finally write the derived data.
    TRY.
        INSERT <ls_derived_data> INTO TABLE <lt_derived_data>.
        io_write_data->write_data(
          EXPORTING
            i_entity     = if_usmdz_cons_entitytypes=>gc_entity_cctr
            it_key       = lt_key
            it_attribute = lt_attribute
            it_data      = <lt_derived_data> ).
      CATCH cx_usmd_write_error.
        "You could transform the exception to ET_MESSAGE_INFO if desired.
    ENDTRY.
  ENDLOOP.
ENDMETHOD.
```

Copyright