PUBLIC

How-To: Create a Custom Handler Class

Applicable Releases:

All

Version 3.0

March 2023

Document History

Document Version	Description
1.0	First official release of this guide with MDG 6.0
1.1	Updates for MDG 6.1
1.2	Updates for MDG 1909 (Jul 2020)
2.0	Layout update (Feb 2023)
3.0	Inclusion of the guide " How-To: Create ERP Suppliers Automatically Using MDG-S" into this guide.

1 Business Scenario

SAP Master Data Governance for Business Partners, Customers or Suppliers (MDG-BP/C/S) provides business processes to find, and maintain business partner, customer, or supplier master data. It supports data governance in a central hub and the distribution to connected operational and business intelligence systems.

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

This How-To Guide describes an example for extending MDG-C or MDG-S by creating and registering a custom handler class if the SAP delivered handler classes do not meet the requirements of your processes. You want to create a custom handler class for use within the MDG application layer. The given scenario is a generic description. Custom handler classes might be relevant for different scenarios.

The example in chapter 5 describes how to change the behavior of the existing MDG-S UI so that it always creates the ERP vendor record, even if the user does not maintain any ERP vendor-specific data (such as general data, company codes or purchasing organizations).

2 Background Information

Custom handler classes must implement the interface `IF_MDG_BS_BP_ACCESS_HANDLER`.

If you want to create a custom handler class with inheritance, you can use the following abstract handler classes as parent:

- `CL_MDG_BS_FND_HANDLER` for the foundation layer `MDG_FND`
- `CL_MDG_BS_ECC_HANDLER` for the application layer `MDG_APPL`

Register your custom handler classes for usage in database table `MDG_BS_BP_HNDLR`. Otherwise, your classes won't be called by MDG's access class.

3 Step by Step Explanation

The following explanation shows you how to create and register a custom handler class.

3.1 Create a custom handler class

Start transaction `SE24` and create a new class in the customer (Z) namespace.

- Name the new class `ZCL_CUSTOM_ECC_HANDLER`.
- Maintain the class description as “My custom application handler”.
- Define the class as “Usual ABAP Class”.
- Decide if the class is Final or not.

Switch to the *Properties* tab and add the `CL_MDG_BS_ECC_HANDLER` superclass. With that you can reuse the common implementations and static variables of both class `CL_MDG_BS_ECC_HANDLER` and its parent class `CL_MDG_BS_FND_HANDLER`.

Save the class. You can also add your custom class to a workbench transport request if you want to use it in other systems according to your system landscape setup.

Since the superclass is abstract, create redefinitions for all methods belonging to the interface `IF_MDG_BS_BP_ACCESS_HANDLER`. The redefinitions themselves can be empty at first. It is enough to implement only those methods that are required to fulfill the needs of your process.

Save, check, and activate the class.

3.2 Register your custom handler class

Switch to the view maintenance transaction `SM30` and select view `V_MDG_BS_BP_HDL`.

Create a new entry with Data Model BP and Class/Interface `ZCL_CUSTOM_ECC_HANDLER`.

Save the new entry. You can also add the registration to a customizing transport request if you want to use it in other systems according to your system landscape setup.

3.3 Verify the usage of your custom handler class

The `ZCL_CUSTOM_ECC_HANDLER` custom handler class is created and registered for the handler processing. To validate its functionality, complete the following steps:

- Set an external breakpoint in the `CL_MDG_BS_ECC_ACCESS` class at the end of the `PROVIDE_HANDLER` method.
- Start the user interface for customer governance or supplier governance.
- In the debugger, display the `GT_HANDLER_ACCESS` attribute of the `CL_MDG_BS_ECC_ACCESS` class.
- Find the `ZCL_CUSTOM_ECC_HANDLER` custom handler class in the `GT_HANDLER_ACCESS` table.

3.4 Add custom logic to your custom handler class

Once your class is defined and registered, add your custom logic in the methods according to your project specific requirements.

3.4.1 Sequence of handler class processing

SAP delivered handler classes will always be processed before your custom class. This behavior is intentional and will never be changed by SAP.

The sequence ensures that your custom handler can overrule the functionality of the SAP delivered handler classes.

3.4.2 Copying SAP delivered code

The SAP delivered classes contain a lot of examples for how to structure custom code. You can surely use this code as a template for your own code. But you should never copy the SAP owned code for overwriting or replacing SAP delivered functionality. This is especially important of you redefine methods without a call to the parent class. If you do this, be aware that the copied SAP code cannot be improved by SAP Notes anymore. SAP cannot provide SAP Notes changing the coding of your custom class. If you do this kind of copying, you must check and copy every SAP Note code change to your custom class. Otherwise, your custom class will run outdated and invalid code, likely causing undesired side-effects.

Your custom handler class shall contain custom code being completely independent from SAP owned code.

4 Performance

The SAP delivered classes contain a lot of reusable methods and attributes. Familiarize yourself with the SAP owned code and attributes / buffers.

Avoid executing expensive processing logic that was called by the SAP's own classes before. An example is reading active data from the database. The SAP-owned handler classes read the complete business partner, including all multiple assignments, customers, and suppliers. The data is stored in the database buffers `GT_BP_DATA_DB` or `GT_ECC_EXTERN_DB`. The buffers contain more tables and fields than supported by the data model BP. If you require data for custom enhancements, check the buffers first. Don't re-read data that is already buffered. Re-read and buffer only those tables and fields that are not covered by the SAP-owned code.

SAP uses SMT mappings to map data from the API format into the staging format and vice versa. SMT simplifies custom enhancements. You can easily add and map custom fields for existing SAP tables and entity types. The negative impact is the runtime required by SMT. If you add a custom table to MDG's data model BP, consider coding the mapping in your custom handler class.

Many methods of the handler interface are supplied with a reference of `IF_USMD_MODEL_EXT`. This reference allows cross reading data that might not be present in the current method but is required for its processing logic. You should never use the method `READ_ENTITY_DATA_ALL`. This method is very expensive since it processes all tables and attributes of data model BP. Instead use `READ_CHAR_VALUE` with providing detailed selection criteria in `IT_SEL`. This limits the data to be processed and thus ensures a short runtime. Also consider setting parameter `IF_NO_FLUSH to ABAP_TRUE` to prevent a flush which is very likely not needed by your custom handler class.

5 Example for Create ERP Suppliers Automatically Using MDG-S

This chapter describes the extension of the MDG Business Partner / Customer / Supplier solution to ensure that ERP Vendor records are always created when users create suppliers in the main user interface for Supplier Governance.

The SAP delivered MDG-S UI configuration is designed in a way that allows the creation of a supplier with or without an ERP vendor record. This supports the different use cases of the supplier. An SRM supplier does not need the ERP vendor data.

The following describes how to change the behavior of the existing MDG-S UI so that it always creates the ERP vendor record, even if the user does not maintain any ERP vendor-specific data (such as general data, company codes or purchasing organizations).

5.1 Technical Background

In the SAP delivered MDG-S UI configuration, the creation of the ERP vendor is triggered by pressing the *New* button on the *ERP Vendors* list. The first vendor created is the so-called "standard assignment". The standard vendor assignment has a fixed assignment category *SUPPL* (this category is actually fixed for all vendor assignments) and a fixed assignment ID *000000000001*.

Since the task is the automated creation of the standard vendor assignment, a suitable place to implement a solution is in the data derivation functionality of a handler class. In that case, it makes sense to create a custom handler class that inherits from the abstract MDG application handler since the objects to be created belong to this software layer. The custom derive functionality needs to create both a multiple assignment entity and an ERP-specific vendor general data entity. A suitable trigger could be the creation of a business partner entity since this happens only once right during the start of the UI application.

5.2 Example Step by Step Explanation

The following steps provide details on how to ensure the creation of an ERP vendor record.

1. Follow the steps in chapter 3 Step by Step Explanation to create your own handler class by inheriting from class **CL_MDG_BS_ECC_HANDLER**. An existing custom handler can be reused, of course.
2. Get familiar with the data derivation implementations of the multiple assignments' handler class **CL_MDG_BS_MLT_ASSGNMNT_HANDLER** and the supplier's handler class **CL_MDG_BS_SUPPL_HANDLER**. Notice that both classes basically already provide the application logic that you need for the scenario and you only need to combine the classes in a different way.
3. As mentioned before, the trigger for creating the ERP vendor is the creation of the business partner during the start of the application. Since the implementation of the multiple assignments' handler method **BUFFER_DERIVED_DATA** already investigates creations of the business partner entity and stores this trigger in the attribute **GV_BPROOT_TASK**, there is no need to do this again in the custom handler.
4. Go to the custom handler class and select the redefinition of method **DERIVE_DATA**. This is the correct place for the creation of the ERP vendor. Your coding needs to:

    a. Ensure that it is only executed for the creation of a single, new business partner of type supplier.

b. Double-check the derive buffer to ensure that the derivation has not happened before by any other handler class or rule service derivation.

c. Prepare the multiple assignment entity type.

d. Prepare the ERP Vendor's general data entity type.

e. Derive both including the update of the local derive buffer.

5. (See the appendix for the complete sample coding.)
6. Save and activate the method.

As the result, the derivation functionality of the custom handler class now creates the ERP vendor records automatically as soon as the UI is started. Run the user interface application for Supplier Governance to prove this. As soon as the UI loaded, an entry in the list *ERP Vendors* is displayed. This record is the derived standard supplier assignment. Navigate into the edit page to check the derived ERP vendor general data values such as the *Account Group*.

5.3 Appendix Example Coding

```
  METHOD if_mdg_bs_bp_access_handler~derive_data.
*! The example shows how to automatically create an ERP Vendor entity.
*
*  It respects the actual hierarchy of entity types by also creating
*  required Multiple Assignment entity type which is the leading parent
*  of an ERP Vendor.
*
*  IMPORTANT NOTE
*  Using the derivation for CREATING entities can cause issues if you
*  also use the BOL Entity Cloner (Copy) functionality. You have to be
*  aware that derivation can create objects that would also be copied.
*  This may lead to error messages, such as "There is already an object
*  with the same key information". If so, you need to decide whether you
*  want to derive or to copy. For more information about how-to control
*  the BOL copy, see the appendix of SAP note 2714704.

    "Since the custom handler is called for all UI configurations
    "and various business activities (like create, change, initial
    "load), it is recommended to limit the automated creation to
    "a) the MDG-S related business activity for creating a supplier
    "b) the exact amount of one single record

    IF mv_process <> 'SUP1'
      OR lines( gt_bproot_task ) <> 1.
      RETURN.
    ENDIF.

    "Furthermore, we want to create the ERP Vendor only once during the
    "start-up of the UI
    FIELD-SYMBOLS <ls_bproot_task> LIKE LINE OF gt_bproot_task.
    READ TABLE gt_bproot_task ASSIGNING <ls_bproot_task> INDEX 1.
    IF sy-subrc <> 0
      OR <ls_bproot_task>-task <> gc_ins.
      RETURN.
    ENDIF.

    "Add the creation of the multiple assignment entity for a standard
    "supplier. The coding is similar to the one in the multiple
    "assignments' handler method DERIVE_STANDARD_ASSIGNMENT.
    DATA:
      ls_bp_mlt_as TYPE /mdgbpx/_s_bp_pp_bp_mlt_as,
      lt_bp_mlt_as TYPE tty_bp_mlt_as_sorted.

    FIELD-SYMBOLS <ls_mlt_as_derived_data> LIKE LINE OF

    gt_mlt_as_derived_data.

    "Check: assignment is not in buffer
    READ TABLE gt_mlt_as_derived_data ASSIGNING <ls_mlt_as_derived_data>
      WITH KEY partner-header-object_instance-bpartner = <ls_bproot_task>-
    bp_id.
      IF sy-subrc EQ 0.
     READ TABLE <ls_mlt_as_derived_data>-mlt_assignments TRANSPORTING NO
       FIELDS WITH KEY assignment_id =
       if_mdg_bs_ecc_bp_constants=>gc_sp_std_mlt_assignment_id partner
       = <ls_bproot_task>-bp_id.
      IF sy-subrc EQ 0.
```

```
        RETURN.
        ENDIF.
    ENDIF. _

    "Prepare assignment data
    ls_bp_mlt_as-assgnm_id = if_mdg_bs_ecc_bp_constants
=>gc_sp_std_mlt_assignment_id.
    ls_bp_mlt_as-bp_header = <ls_bproot_task>-bp_id.
    ls_bp_mlt_as-as_type= if_mdg_bs_ecc_bp_constants
=>gc_sp_mlt_assignment_cat.
    ls_bp_mlt_as-standard   = abap_true.

    "Add the creation of the ERP vendor general data entity for a
    "standard supplier next.  The coding is similar to the one in the
    "supplier's handler method DERIVE_SUPPLIER.

    DATA:
      lo_exception TYPE REF TO cx_mdg_bs_ecc_bp_obj_integr,
      ls_bp_vengen TYPE /mdgbpx/_s_bp_pp_bp_vengen,
      ls_message        TYPE usmd_s_message,
      lt_bp_vengen TYPE tty_bp_vengen_sorted,
      lv_bp_group   TYPE bu_group.

    "Check: general data is not in buffer (inbound!)
    IF <ls_mlt_as_derived_data> IS ASSIGNED.
      READ TABLE <ls_mlt_as_derived_data>-vendors TRANSPORTING NO FIELDS WITH KEY
      assignment_id = if_mdg_bs_ecc_bp_constants=>gc_sp_std_mlt_assignment_id.
      IF sy-subrc EQ 0.
        RETURN.
      ENDIF.
    ENDIF.

    "Prepare supplier entity keys
    ls_bp_vengen-bp_header = <ls_bproot_task>-bp_id.
    ls_bp_vengen-assgnm_id =
if_mdg_bs_ecc_bp_constants=>gc_sp_std_mlt_assignment_id.

    "Derive account group and ID.
    me->derive_bp_group(
      EXPORTING
        io_model             = io_model
        iv_bp_id             = <ls_bproot_task>-bp_id
      IMPORTING
        ev_bp_group          = lv_bp_group
      CHANGING
        ct_message_info = ct_message_info ).
    IF lv_bp_group IS INITIAL.
      "There's no (default) grouping for the BP. Without the grouping,
      "the derivation cannot proceed!
      RETURN.
    ENDIF.
    TRY.
        ls_bp_vengen-ktokk = cl_mdg_bs_ecc_bp_services
=>get_supplier_account_for_group( lv_bp_group ).
        ls_bp_vengen-lifnr = cl_mdg_bs_ecc_bp_services
=>get_supplier_number(
                                iv_account_group = ls_bp_vengen-ktokk
                                iv_bp_group           = lv_bp_group
                                iv_partner            = <ls_bproot_task>-bp_id ).
      CATCH cx_mdg_bs_ecc_bp_obj_integr INTO lo_exception.
```

```
        CLEAR ls_message.
        ls_message-msgty = 'E'.

        ls_message-msgid = lo_exception->if_t100_message~t100key-msgid.
        ls_message-msgno = lo_exception->if_t100_message~t100key-msgno.

        ls_message-msgv1 = lo_exception->msgv1.

        APPEND ls_message TO ct_message_info.
        RETURN.

    ENDTRY.

    "Derive sub-range and plant relevance.

    ls_bp_vengen- ltsna =
cl_mdg_bs_ecc_bp_services=>is_sub_range_relevant( ls_bp_vengen-
ktokk ).
     ls_bp_vengen- werkr =
cl_mdg_bs_ecc_bp_services=>is_plant_level_relevant( ls_bp_vengen-
ktokk ).

    "Derive trading partner
    ls_bp_vengen-vbund = me->derive_trading_partner(

                         io_model = io_model
                         iv_bp_id = <ls_bproot_task>-bp_id ).

    "WRITE the derived data
    INSERT ls_bp_mlt_as INTO TABLE lt_bp_mlt_as.

    io_write_data->write_data( i_entity =
if_mdg_bs_ecc_bp_constants =>gc_ma_model_entity

it_data   = lt_bp_mlt_as ).
    me->if_mdg_bs_bp_access_handler~buffer_derived_data(

      EXPORTING
        io_model    = io_model

        iv_entity   =
if_mdg_bs_ecc_bp_constants=>gc_ma_model_entity
        it_data_ins = lt_bp_mlt_as ).

    INSERT ls_bp_vengen INTO TABLE lt_bp_vengen.

    io_write_data- >write_data( i_entity =
if_mdg_bs_ecc_bp_constants=>gc_sp_model_entity- general_data

                                it_data   = lt_bp_vengen ).
    me->if_mdg_bs_bp_access_handler~buffer_derived_data(

      EXPORTING
        io_model    = io_model

        iv_entity   =
if_mdg_bs_ecc_bp_constants=>gc_sp_model_entity- general_data

        it_data_ins = lt_bp_vengen ).

  ENDMETHOD.
```

6 Additional Information

6.1 Further Reading

6.1.1 Information on SAP MDG on SAP S/4HANA

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

6.1.2 SAP Roadmap Explorer

- Please see the roadmap for SAP Master Data Governance

6.1.3 Related Information

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

6.2 SAP Notes

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

Note	Description
2221398	MDG-BP/C/S/CA: (Un-)Supported Fields in Data Model BP
2847807	MDG-BP/C/S/CA: Usage of MDG Tools and Processes
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
1619534	How to Create, Enhance and Adapt FPM Applications
1637249	MDG: Information for efficient message processing
2105467	MDG Performance
2561461	Scope of support for SAP Master Data Governance (MDG)