



PUBLIC

How-To: Extend the MDG Business Partner – Filter an Input Help for a Business System

Applicable Releases:

From EHP6 for SAP ERP 6.0 and from S/4HANA 1511

Version 2.0

April 2024

Document History

Document Version	Description
1.0	First official release of this guide
1.1	August 2016
2.0	Update Layout (April 2024)

1	BUSINESS SCENARIO.....	4
2	TECHNICAL BACKGROUND.....	5
2.1	Business Systems.....	5
2.2	Input Helps	5
2.3	Value Mapping	5
2.4	Key Mapping	6
3	STEP BY STEP IMPLEMENTATION GUIDE.....	7
3.1	Create a New User Parameter	7
3.2	Add a Field “Business System” to the Business Partner Data	7
3.3	Set the Parameter that Defines the Business System According to the Current User	7
3.4	Filter the Input Help of the Field “Company Code”	8
3.5	Filter the Input Help of the Field “Purchasing Organization”	8
3.6	Result	9
4	APPENDIX	10
4.1	BAdI Implementation IF_EX_USMD_RULE_SERVICE2~DERIVE.....	10
4.2	Implementation ZCL_BS_SP_GUIBB_COMPANY_CODES → FILTER_COMPANY_CODES	11
4.3	Implementations in ZCL_BS_SP_GUIBB_PURCH_ORG	12
4.3.1	CREATE_STRUCT_RTTI	12
4.3.2	GET_DEFINITION.....	12
4.4	Implementations in ZCL_BS_SP_GUIBB_PURCH_ORGS.....	13
4.4.1	HANDLE_PHASE_1.....	13
4.4.2	OVS_HANDLE_PHASE_2.....	13
4.4.3	OVS_OUTPUT_EKORG	13
4.4.4	OVS_HANDLE_PHASE_3.....	15
5	ADDITIONAL INFORMATION	16
5.1	Further Reading	16
5.1.1	Information on SAP MDG on SAP S/4HANA	16
5.1.2	SAP Roadmap Explorer	16
5.1.3	Related Information	16
5.2	SAP Notes.....	16

1 Business Scenario

SAP Master Data Governance (MDG) provides business processes to find, create, change, and mark master data for deletion. It supports the governance of master data 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.

MDG offers change request (CR)-based processing of master data with integrated workflow, staging, approval, activation, and distribution.

This How-To Guide describes the extension of the MDG Business Partner / Customer / Supplier solution for filtering input helps according to business systems.

Business systems are the targets or clients of the MDG hub. Users maintaining master data on the MDG hub are sometimes the experts from the different business systems. Since the MDG hub usually feeds more than one business system, it often has a larger scope of master data and related customizing values. This larger scope might confuse the user into maintaining invalid master data.

The requirement is to include the business system into the business partner data. It shall be set according to the current user working with the master data. Furthermore, it shall be used as a filter criterion for input helps (for example, to limit the amount of company codes and purchasing organizations the user may define).

The given scenario is a complex description. It requires general knowledge about SAP systems, BAdI implementation, and UI extension. Before following the instructions in this guide, we recommend you study the following guides on [How-to Information for Business Partner](#):

- [How-to Guide: Overview of MDG-BP/C/S](#)
- [How-to Guide: Create or Redefine a UI Feeder Class](#)
- [How-to Guide: Extend MDG-S / MDG-C Data Model by a New Field \(Reuse Option\)](#)

2 Technical Background

2.1 Business Systems

Business systems as clients of the MDG hub system can be defined in the Data Replication Framework (DRF) customizing. The customizing can be used for a value help when you maintain the business system as field value later on.

There is currently no standard solution available to link the current user with a business system.

The following options are possible:

1. The creation of a user parameter to assign the business system to a user.
2. The creation of a custom table containing the link between business systems and users.

The step-by-step implementation guide uses the first option as example.

2.2 Input Helps

The UIs for MDG-C and MDG-S currently use different techniques for input helps. Here are some examples:

- Value helps related to domain fixed values respectively check tables
- Search helps for the data dictionary
- Object Value Selector (OVS) search helps (refer to the [Floorplan Manager for Web Dynpro ABAP](#) for more details).

Web Dynpro and FPM try to determine the desired input help in a generic way; for example, domain fixed values are displayed as value help automatically and search helps for the data dictionary are added to UI fields automatically.

It is still possible to use the feeder class method **GET_DEFINITION** to redefine these settings. You can use this class, for example to set a different DDIC search help to the current one, or to define a link to an OVS for complex searches.

The fixed values of value helps are determined by the feeder's method **GET_ATTR_VALUE_SET**. The search results of an OVS are determined in the second phase of the OVS, usually in methods **HANDLE_PHASE_2** of the interface **IF_FPM_GUIBB_OVS** or in **OVS_HANDLE_PHASE_2** of the class **CL_GUIBB_BOL_BASE**. These methods allow the creation of specific search results and thus are possible options for redefining what search results should be. The result of search helps for the data dictionary cannot be redefined since these search helps are called in a fully generic way. The step-by-step implementation guide shows both an example for redefining a value help and OVS.

2.3 Value Mapping

Value mapping is maintained in the MDG specific customizing. Call transaction **MDGIMG** and then navigate to **Master Data Governance → General Settings → Value Mapping**.

Item **Overview** explains the usage of value mapping in general. The recommendations given are valid for the business partner, too. If data is distributed using the enterprise service, you must maintain value mapping both on the MDG hub and client system. The message being sent between the systems contains global codes. If data is distributed using ALE IDocs, you only have to maintain value mapping on the MDG hub.

Item **Maintain Value Mapping** allows the creation of value mapping for specific business systems. To enable the filtering of input helps according to values of the business system, you must maintain suitable entries.

The step-by-step implementation guide shows how these values are accessed during the data maintenance in the UI.

2.4 Key Mapping

Key mapping is maintained in the MDG transaction **MDG_KM_MAINTAIN**. Key mapping is used for the keys of business partners, customer and vendors as well as sub-objects such as purchasing organizations or sales areas.

3 Step by Step Implementation Guide

3.1 Create a New User Parameter

The following steps are required to establish a direct link of a business system to a user with the help of a user parameter.

1. Run transaction **SM32** for table **TPARA**.
2. Create a new user parameter **ZMDG_SYSTEM**. Enter a meaningful short text, for example: "MDG Default Business System".
3. Run transaction **SU01** for the user(s) that shall be linked with a business system.
4. Switch to tab **Parameters**.
5. Add parameter **ZMDG_SYSTEM** and set a suitable value.
6. Save the changes.

3.2 Add a Field "Business System" to the Business Partner Data

Create the business system as new field within the business partner data, using the following [How-to Guide: Extend MDG-S / MDG-C Data Model by a New Field \(Reuse Option\)](#). Consider the following proposals for names and data types.

- In the CI Includes, name the field **ZZ_SYSTEM** using the data element **MDG_BUSINESS_SYSTEM** respectively **BAPIUPDATE**.
- In data model BP name the field **ZZSYSTEM** using the data element **MDG_BUSINESS_SYSTEM**.

Once the how-to guide is completed a new field for the business system is available in the business partner's central data entity. The field already has a search help due to the chosen data element. Currently the search help returns all business systems that are available in the MDG hub.

3.3 Set the Parameter that Defines the Business System According to the Current User

The following steps are required to pre-define the business system according to the current user. This is realized by implementing the USMD derive BAdI.

1. Run transaction **MDGIMG**.
2. Navigate to **Master Data Governance → General Settings → Data Quality and Search → Business Add-Ins → Derivations Across Entity Types**.
3. Create a new BAdI implementation or re-use an existing one. Ensure that the BAdI uses data model **BP** as filter.
4. Switch to the implementation class.
5. Implement method **IF_EX_USMD_RULE_SERVICE2~DERIVE** in so that it sets the user parameter into the new field for the business system. Refer to the source code in the chapter 4 Appendix.
6. **Save** and **Activate** your changes.

3.4 Filter the Input Help of the Field “Company Code”

The following steps are required to filter the results of a common value help based on MDG value mapping for a business system. The example is a filter for the ERP vendor company code field. It requires maintained value mapping for global data type **BUKRS** for the related business system.

1. Create a redefinition of the company code list feeder **CL_BS_SP_GUIBB_COMPANY_CODES** as described in [How-to Guide: Create or Redefine a UI Feeder Class](#).
 - Name the class **ZCL_BS_SP_GUIBB_COMPANY_CODES**.
 - Assign it to a customized version of list UIBB **BS_SP_COMPANY_CODES**.
2. Get familiar with class **CL_BS_SP_GUIBB_COMPANY_CODES**. As mentioned, the company code field currently uses a value help. Related values are determined by method **GET_ATTR_VALUE_SET**. This method already calls a filter method for company codes: **FILTER_COMPANY_CODES**. Since this method is public, it can be redefined to add the value mapping filter according to the current business system.
3. Navigate to the new feeder class **ZCL_BS_SP_GUIBB_COMPANY_CODES**.
4. Create a redefinition for method **FILTER_COMPANY_CODES**. Refer to chapter 4 Appendix for the complete source code.
5. **Save** and **Activate** your changes.

3.5 Filter the Input Help of the Field “Purchasing Organization”

The following steps are required to filter the results of a search help (OVS) based on MDG key mapping for a business system. The example is a filter for the ERP vendor purchasing organization field. It includes switching the purchasing organization field from a dropdown list-box to an input field with text field to enable the OVS search help.

1. First change the existing purchasing organization field so that it is an input field with related text field using an OVS search help.
 - Redefine the purchasing organization form feeder **CL_BS_SP_GUIBB_PURCH_ORG** as described in [How-to Guide: Create or Redefine a UI Feeder Class](#).
 - Name the class **ZCL_BS_SP_GUIBB_PURCH_ORG**. Get familiar with class **CL_BS_SP_GUIBB_PURCH_ORG**.
 - As mentioned, the purchasing organization field is currently a dropdown list-box. A change to input field with text field requires the creation of the related text field in method **CREATE_STRUCT_RTTI**. The source code is already part of the feeder but commented. Use the code to create a redefinition of the method in the custom feeder. Refer to chapter 4 Appendix for the complete source code.
 - The assignment of an OVS search help must be done explicitly. The valid method for this task is **GET_DEFINITION**. The feeder links an ABAP DDIC search help for the purchasing organization. You must overwrite this search help with a redefinition. Refer to chapter 4 Appendix for the complete source code.
 - **Save** and **Activate** your changes.
2. Create a customizing for form UIBB **BS_SP_PURCH_ORG**.

- Exchange the existing feeder class with class **ZCL_BS_SP_GUIBB_PURCH_ORG**.
 - Change the display type of field purchasing organization from Drop Down to Input Field.
 - Add the text field **EKORG__TEXT** in the same row in columns I – P as Input Field without label.
 - **Save** the changes.
3. Implement the OVS search help. Refer to the [Floorplan Manager for Web Dynpro ABAP](#) for details about the OVS search help.
- Create a redefinition of the purchasing organization code list feeder **CL_BS_SP_GUIBB_PURCH_ORGS** as described in [How-to Guide: Create or Redefine a UI Feeder Class](#).
 - Name the class **ZCL_BS_SP_GUIBB_PURCH_ORGS**. Assign it to a customized version of list **UIBB_BS_SP_PURCH_ORGS**.
 - *Optional:* Redefine OVS phase 1 to show a custom input screen for the search help. The search screen shows the current business system as query parameter. This is done in method **HANDLE_PHASE_ONE**. Refer to chapter 4 Appendix for the complete source code.

- Redefine OVS phase 2 to build the search result. The generic method **OVS_HANDLE_PHASE_2** is re-implemented to call a new method **OVS_OUTPUT_EKORG** being responsible for delivering the search result. In this method it is possible to re-use the existing determination of purchasing organizations. Refer to the appendix for the complete source code.
- *Optional:* Since the purchasing organization field is a key field it makes sense to immediately trigger a UI round-trip as soon as the user has selected a value in the search help. This is possible in OVS phase 3 by redefining **OVS_HANDLE_PHASE_3**. Refer to the appendix for the complete source code.
- **Save** and **Activate** your changes.

3.6 Result

Business partner data within SAP MDG is now enhanced with a field Business System. The field's value is set during data derivation according to a user parameter. Still the field is changeable during data maintenance. It serves as basis for reducing the result list of both a value help and OVS search help. The results of the value help for company codes are filtered according to existing value mapping. The results of the OVS search help for purchasing organizations are filtered according to existing key mapping.

4 Appendix

4.1 BAdI Implementation IF_EX_USMD_RULE_SERVICE2~DERIVE

```
DATA:
  lr_t_ins      TYPE REF TO data,
  ls_parameter  TYPE bapiparam,
  lt_attribute  TYPE usmd_ts_fieldname,
  lt_parameter  TYPE TABLE OF bapiparam,
  lt_return     TYPE TABLE OF bapiret2.

FIELD-SYMBOLS:
  <ls_data> TYPE any,
  <lt_ins>   TYPE INDEX TABLE,
  <lv_system> TYPE any.

"According to the requirement it is sufficient to pre-define the business system once. A perfect point
"in time is the very first creation of central BP data since the business system field is part of it.

CLEAR lr_t_ins.
io_changed_data->read_data(
  EXPORTING
    i_entity      = 'BP_CENTRL'
    i_struct      = io_model->gc_struct_key_attr
  IMPORTING
    er_t_data_ins = lr_t_ins ).
ASSIGN lr_t_ins->* TO <lt_ins>.
IF <lt_ins> IS NOT ASSIGNED
  OR <lt_ins> IS INITIAL.
  RETURN.
ELSE.
  READ TABLE <lt_ins> ASSIGNING <ls_data> INDEX 1.
  ASSIGN COMPONENT 'ZZSYSTEM' OF STRUCTURE <ls_data> TO <lv_system>.
  IF <lv_system> IS NOT ASSIGNED.
    RETURN.
  ENDIF.
ENDIF.

"The user parameter is obtained by a common function module. The parameter shall be transferred to
"central data only if it has a value.

CALL FUNCTION 'BAPI_USER_GET_DETAIL'
  EXPORTING
    username = sy-uname
  TABLES
    parameter = lt_parameter
    return    = lt_return.
CLEAR ls_parameter.
READ TABLE lt_parameter INTO ls_parameter
  WITH KEY parid = 'ZMDG_SYSTEM'.
IF ls_parameter-parva IS INITIAL.
  RETURN.
ENDIF.

"Using the write interface of the derive method, the new value for the business system field is
"finally written into the business partner central data entity.

CLEAR: lt_attribute.
<lv_system> = ls_parameter-parva.
APPEND 'ZZSYSTEM' TO lt_attribute.

"write the data
TRY.
  io write_data->write_data(
    EXPORTING
      i_entity      = 'BP_CENTRL'
      it_attribute  = lt_attribute
      it_data       = <lt_ins> ).
  CATCH cx_usmd_write_error. " Error During Write Access
  RETURN.
ENDTRY.
```

4.2 Implementation ZCL_BS_SP_GUIBB_COMPANY_CODES → FILTER_COMPANY_CODES

"The following declarations are needed.

```
DATA:
    lo_entity          TYPE REF TO cl_crm_bol_entity,
    lo_value_mapping    TYPE REF TO cl_mdg_value_mapping_api,
    lt_codes            TYPE cl_mdg_value_mapping_api=>tt_ext_codetab,
    ls_code            LIKE LINE OF lt_codes,
    lt_company         TYPE wdr_context_attr_value_list,
    ls_company         LIKE LINE OF lt_company,
    lt_obj_data        TYPE cl_mdg_value_mapping_api=>tt_obj_data,
    lv_system          TYPE string.
```

"Since it is important to run through the parent class' code, implement a super call.

```
super->filter_company_codes(
    EXPORTING
        io_access      = io_access
        iv_object_name = iv_object_name
    CHANGING
        ct_company     = ct_company ).
```

```
IF ct_company IS INITIAL.
    RETURN.
ENDIF.
```

"Check if the business system is set. The field is part of the business partner central data.

"Parameter io_access is actually a valid BOL entity. It can be used to navigate from the company

"code to the root entity that contains the central data in order to obtain the business system

"value.

"cast io_access to a BOL entity.

```
IF io_access IS NOT BOUND.
    RETURN.
ENDIF.
TRY.
    lo_entity ?= io_access.
    CATCH cx_sy_move_cast_error.
        RETURN.
ENDTRY.
```

"navigate to the BP central data that is part of the root entity

```
TRY.
    lo_entity = lo_entity->get_root( ).
    CATCH cx_crm_genil_model_error.
        RETURN.
ENDTRY.
```

"get the business system value

```
lv_system = lo_entity->get_property_as_string( iv_attr_name = 'ZZ_SYSTEM' ).
IF lv_system IS INITIAL.
    RETURN.
ENDIF.
```

*"Access the value mapping to get the list of defined values for the business system. Use the
"value mapping API to get the company code values.*

```
CREATE OBJECT lo_value_mapping.
"prepare the read access to value mapping
CLEAR lt_obj_data.
APPEND INITIAL LINE TO lt_obj_data ASSIGNING <ls_obj_data>.
"company code is a data element
<ls_obj_data>-object = 'DTEL'.
"object name (global data type) as maintained in customizing
<ls_obj_data>-obj_name = 'BUKRS'.
"list ID as maintained in customizing
<ls_obj_data>-list_id = 'BUKRS'.
"read the value mapping - the needed result is the external code table
```

```

TRY.
  lo_value_mapping->read(
    EXPORTING
      it_obj_data          = lt_obj_data
    IMPORTING
      et_ext_codetab       = lt_codes ).
  CATCH cx_mdg_code_mapping cx_mdg_value_mapping_api.
  RETURN.
ENDTRY.
"Filter the given list of company codes according to the value mapping results. Only company
"codes existing in value mapping shall remain selectable. If the value mapping did not return
"any company codes, the given values are not touched.
CLEAR lt_company.
LOOP AT lt_codes INTO ls_code
  WHERE list_agency_id = lv_system.
  READ TABLE ct_company INTO ls_company
    WITH KEY value = ls_code-internal_code.
  IF sy-subrc EQ 0.
    APPEND ls_company TO lt_company.
  ENDIF.
ENDLOOP.
IF lt_company IS NOT INITIAL.
  ct_company = lt_company.
ENDIF.

```

4.3 Implementations in ZCL_BS_SP_GUIBB_PURCH_ORG

4.3.1 CREATE_STRUCT_RTTI

```

DATA:
  lt_components TYPE cl_abap_structdescr=>component_table.
FIELD-SYMBOLS:
  <ls_component> LIKE LINE OF lt_components.
"call parent first
super->create_struct_rtti( ).
"get current UI components
lt_components = me->mo_struct_rtti->get_components( ).
"Add Purchasing Organization text. The field value is set by method get_entity_data.
APPEND INITIAL LINE TO lt_components ASSIGNING <ls_component>.
<ls_component>-name = 'EKORG_TEXT'.
<ls_component>-type = cl_abap_elemdescr=>get_string( ).

"update the components
me->mo_struct_rtti = cl_abap_structdescr=>create( lt_components ).

```

4.3.2 GET_DEFINITION

```

FIELD-SYMBOLS:
  <ls_field_description> LIKE LINE OF et_field_description.

"call parent first
super->if_fpm_gui_bb_form~get_definition(
IMPORTING
  es_message          = es_message
  eo_field_catalog    = eo_field_catalog
  et_field_description = et_field_description
  et_action_definition = et_action_definition
  et_special_groups   = et_special_groups
  ev_additional_error_info = ev_additional_error_info
  et_dnd_definition   = et_dnd_definition ).
"Redefine the SAP standard definition for the purchasing organization.
"Add the current class as OVS class and delete the DDIC search help.
READ TABLE et_field_description ASSIGNING <ls_field_description>
  WITH KEY name = 'EKORG'.
IF <ls_field_description> IS ASSIGNED.
  CLEAR <ls_field_description>-ddic_shlp_name.
  <ls_field_description>-ovs_name = me->class_name.
ENDIF.

```

4.4 Implementations in ZCL_BS_SP_GUIBB_PURCH_ORGS

4.4.1 HANDLE_PHASE_1

```
TYPES:
  BEGIN OF ts_input,
    system TYPE mdg_business_system,
  END OF ts_input.

DATA:
  ls_input TYPE ts_input.
"Only field purchasing organization shall get a specific search input.
CASE iv_field_name.
  WHEN 'EKORG'.
    "get defined business system to pre-set the search
    IF me->mo_entity IS BOUND.
      ls_input-system = me->mo_entity->get_root( )->get_property_as_string( 'ZZ_SYSTEM' ).
    ENDIF.
    "set input structure and transfer values
    io_ovs_callback->set_input_structure(
      EXPORTING
        input = ls_input ).
  WHEN OTHERS.
    super->if_fpm_gui_bb_ovs~handle_phase_1(
      EXPORTING
        iv_field_name = iv_field_name
        io_ovs_callback = io_ovs_callback ).
ENDCASE.
```

4.4.2 OVS_HANDLE_PHASE_2

```
"Only field purchasing organization shall get a specific search result.
CASE iv_field_name.
  WHEN 'EKORG'.
    me->ovs_output_ekorg(
      EXPORTING
        ir_query_parameter = ir_query_parameter
      IMPORTING
        er_output = er_output ).
  WHEN OTHERS.
    super->ovs_handle_phase_2(
      EXPORTING
        iv_field_name = iv_field_name
        ir_query_parameter = ir_query_parameter
        io_access = io_access
      IMPORTING
        er_output = er_output
        ev_table_header = ev_table_header
        et_column_texts = et_column_texts ).
ENDCASE.
```

4.4.3 OVS_OUTPUT_EKORG

```
TYPES:
  BEGIN OF ts_input,
    system TYPE mdg_business_system,
  END OF ts_input.

TYPES:
  BEGIN OF ts_result,
    ekorg TYPE ekorg,
    text TYPE ekotx,
  END OF ts_result.

TYPES:
  tt_result TYPE SORTED TABLE OF ts_result WITH UNIQUE KEY ekorg.

DATA:
  lo_key_mapping TYPE REF TO if_mdg_id_matching_api_bs,
  ls_key_mapping TYPE mdg_s_get_matching_easy_bs,
  ls_purch_org TYPE wdr_context_attr_value,
  ls_result TYPE ts_result,
  lt_key_mapping TYPE mdg_t_get_matching_easy_bs,
  lt_purch_orgs TYPE wdr_context_attr_value_list,
```

```

lt_system_ids TYPE mdg_t_business_system_id_range,
lv_own_system TYPE mdg_fnd_business_system_id,
lv_system     TYPE mdg_business_system.
FIELD-SYMBOLS:
<ls_input>    TYPE ts_input,
<lt_result>   TYPE tt_result,
<ls_system_id> TYPE mdg_s_business_system_id_range.
"It is mandatory that er_output is returned to prevent a short dump.
CREATE DATA er_output TYPE tt_result.
ASSIGN er_output->* TO <lt_result>.
"First re-use the existing coding to get the current list of purchasing organizations. This
ensures that the filter for authority checks and existing data is processed.
me->get_attr_value_set(
  EXPORTING
    io_access      = me->mo_entity
    iv_attr_name   = 'EKORG'
  IMPORTING
    et_value_set   = lt_purch_orgs ).
IF lt_purch_orgs IS INITIAL.
  RETURN.
ENDIF.
LOOP AT lt_purch_orgs INTO ls_purch_org.
  ls_result-ekorg = ls_purch_org-value.
  ls_result-text  = ls_purch_org-text.
  INSERT ls_result INTO TABLE <lt_result>.
ENDLOOP.

"Now determine if there is a business system that should be used for filtering. Since the
usage of OVS phase 1 is optional, don't rely on it.
IF ir_query_parameter IS BOUND.
  ASSIGN ir_query_parameter->* TO <ls_input>.
  lv_system = <ls_input>-system.
ELSE.
  lv_system = me->mo_entity->get_root( )->get_property_as_string( 'ZZ_SYSTEM' ).
ENDIF.
IF lv_system IS INITIAL.
  RETURN.
ENDIF.
"get the own business system
lv_own_system = cl_mdg_ca_get_own_bus_sys=>get_local_business_system( ).
IF lv_own_system IS INITIAL.
  RETURN.
ENDIF.
"get an instance of the key mapping API
cl_mdg_ukm=>get_id_matching_api_instance(
  IMPORTING
    eo_mdg_id_matching_api = lo_key_mapping ).
IF lo_key_mapping IS NOT BOUND.
  RETURN.
ENDIF.
"prepare the key mapping query
CLEAR lt_system_ids.
APPEND INITIAL LINE TO lt_system_ids ASSIGNING <ls_system_id>.
<ls_system_id>-business_system_id = lv_own_system.
"get key mapping
CLEAR lt_key_mapping.
TRY.
  lo_key_mapping->query_objects(
    EXPORTING
      iv_object_type_code      = cl_mdg_ukm=>c_type_code_ekorg
      it_business_system_id_range = lt_system_ids
    IMPORTING
      et_mapping_groups        = lt_key_mapping ).
  CATCH cx_mdg_missing_id_data cx_mdg_otc_idm_error
        cx_mdg_id_matching_bs cx_mdg_idsc_invalid
        cx_mdg_missing_input_parameter.
  RETURN.
ENDTRY.
"Analyze the key mapping. If entries exist, return only the ones related to the target source
system. Other entries should not be displayed to the user.
IF lt_key_mapping IS NOT INITIAL.
  CLEAR <lt_result>.
  LOOP AT lt_key_mapping INTO ls_key_mapping.
    "Check first if the key mapping purchasing organization is in the current list
    CLEAR ls_purch_org.

```

```

READ TABLE lt_purch_orgs INTO ls_purch_org
  WITH KEY value = ls_key_mapping-search_key-identifier_key-id_value.
IF sy-subrc NE 0.
  "Not found means not to use this purchasing organization.
  CONTINUE.
ENDIF.
"Check second if there is key mapping for the given business system.
READ TABLE ls_key_mapping-matching_objects TRANSPORTING NO FIELDS
  WITH KEY business_system_id = lv_system.
IF sy-subrc NE 0.
  "Not found means not to use this purchasing organization.
  CONTINUE.
ENDIF.
"The current purchasing organization is a valid one. Add it to the result list.
CLEAR ls_result.
ls_result-ekorg = ls_purch_org-value.
ls_result-text = ls_purch_org-text.
INSERT ls_result INTO TABLE <lt_result>.
ENDLOOP.
ENDIF.

```

4.4.4 OVS_HANDLE_PHASE_3

```

"Call parent first.
super->ovs_handle_phase_3(
  EXPORTING
    iv_field_name = iv_field_name
    ir_selection  = ir_selection
  IMPORTING
    et_field_value = et_field_value
    eo_fpm_event   = eo_fpm_event ).
"Ensure a UI round-trip for the purchasing organization.
IF iv_field_name EQ 'EKORG'.
  eo_fpm_event = cl_fpm_event=>create_by_id( cl_fpm_event=>gc_event_refresh ).
ENDIF.

```

5 Additional Information

5.1 Further Reading

5.1.1 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](#)
- Try SAP Master Data Governance on S/4HANA on the SAP Cloud Appliance Library: [S/4HANA 2022 FPS1](#)
- Learn more: [Latest Release](#) | [Help Portal](#) | [How-to Information](#) | [Key Presentations](#)

5.1.2 SAP Roadmap Explorer

- Please see the [roadmap for SAP Master Data Governance](#)

5.1.3 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](#)

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	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
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
3374711	Functional restrictions in MDG for Business Partner / Customer / Supplier on SAP S/4HANA 2023
3043582	MDG Customer Connection 2020
3194967	MDG Customer Connection 2021 for S/4HANA 2022

3311039	MDG Customer Connection 2023
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)
1637249	MDG: Information for efficient message processing