PUBLIC

How-To: Create or Redefine a UI Feeder Class

Former title: Extend the Business Partner – User Interface

Applicable Releases:

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

Version 2.0

July 2023

Document History

Document Version	Description
1.0	First official release of this guide with MDG 6.0
1.1	Updates for MDG 8.0
1.2	Updates for MDG 1909
2.0	Layout update

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 how to extend the MDG Business Partner / Customer / Supplier solution by either creating or redefining a custom feeder class for the user interface. It explains how this feeder class is added to an existing User Interface Building Block (UIBB).

2 Background Information

The SAP predefined UI feeder classes are not enough for your processes. You need to create a custom feeder class. You can use this class within the related UIBBs.

The requirement is the creation of a custom feeder class that you can use to extend the pre-defined functionality of the SAP standard list feeder class for the company code data of ERP vendors.

General details about FPM feeder classes are available in the FPM Cookbook on SDN. Custom feeder classes must inherit from the MDG business partner feeder classes to ensure a valid functionality.

New feeder classes that add new functionality or that either modify or replace existing SAP standard functionality must inherit from the following classes:

- `CL_BS_BP_GUIBB_FORM` for form UIBBs
- `CL_BS_BP_GUIBB_LIST` for list UIBBs.

New feeder classes that extend existing SAP standard functionality may inherit directly from the affected feeder class.

Every UIBB requires a link to its feeder class. This link is set during the configuration of the UIBB. For existing SAP UIBBs, it is possible to change the feeder class in customizing mode.

3 Step by Step Explanation

3.1 Create a custom feeder class

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

- Name the new class `ZCL_BS_CU_GUIBB_GENERAL_DATA`.
- Maintain the class description as *My customer general data feeder*.
- Define the class as *Usual ABAP Class*.
- Decide if the class is *Final* or not.

Switch to the *Properties* tab and add the `CL_BS_CU_GUIBB_GENERAL_DATA` superclass. This is required since the example is going to re-define some of the SAP owned logic and UIBB configurations.

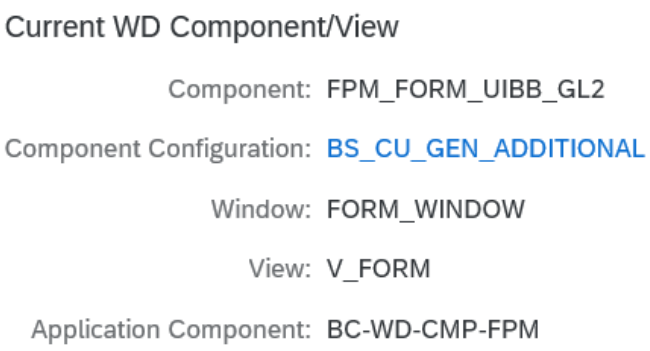
Save, check, and activate the class. You can also add your custom class to a workbench transport request if you want to use it in other systems according to your system landscape setup.

3.2 Use a custom feeder class

Any UIBB requires a feeder class for its functional correctness. The following steps describe how-to replace an SAP owned feeder class in an SAP delivered UIBB configuration. You create a customizing for the UIBB that adds your custom feeder class. There are multiple options for creating the customizing of UIBBs. Refer to the FPM Cookbook for further details.

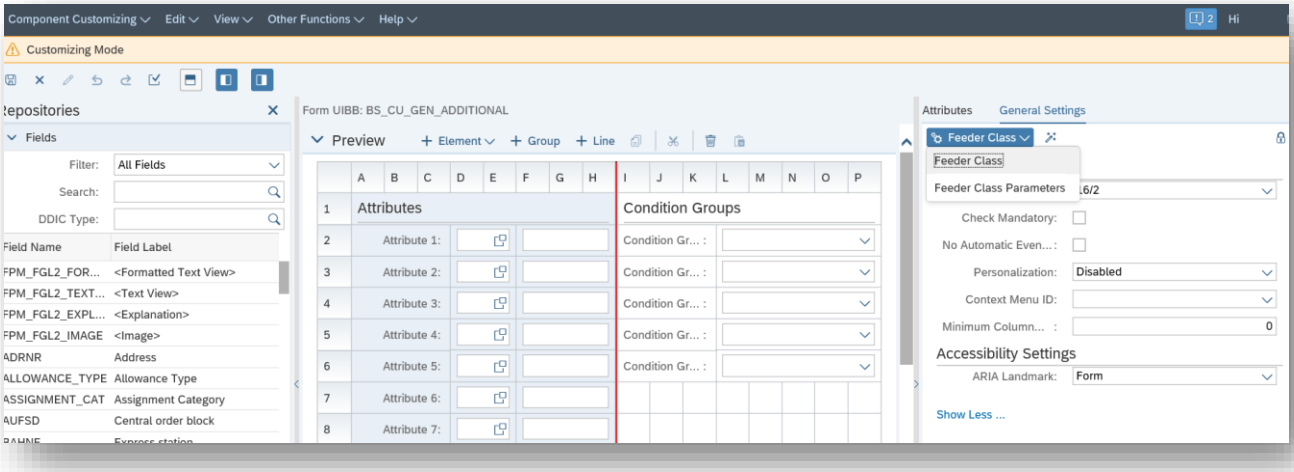
Ensure that user parameter `FPM_CONFIG_EXPERT` is set to *A* in your user profile.

Start the MDG-C UI and navigate to the ERP Customer edit page. Locate the UIBB for *ERP Customer: Additional Data*. Right-click on the UIBB and choose *Technical Help*. You will see a pop-up showing technical details of the current page. Follow the link to the UIBB configuration `BS_CU_GEN_ADDITIONAL`.



A new browser window displays the SAP delivered UIBB configuration *BS_CU_GEN_ADDITIONAL – ERP Customer: Additional Data*. In the top menu, choose *Other Functions* and *Create Customizing* respectively *Edit Customizing*.

Add a description for your newly created customizing. You can also add your customizing to a customizing transport request if you want to use it in other systems according to your system landscape setup.



In the customizing mode of the UIBB, choose *General Settings* → *Feeder Class* → *Feeder Class* as shown in the figure. This opens a pop-up where you replace the current feeder class with your custom one. You can also change feeder class parameters if desired. Finally save your customizing.

3.3 Verify the usage of your custom feeder class

The `ZCL_BS_CU_GUIBB_GENERAL_DATA` custom feeder class is created and registered for the UIBB *ERP Customer: Additional Data*. To validate its functionality, complete the following steps:

- Redefine any method (e.g., `GET_ENTITY_DATA`) in `ZCL_BS_CU_GUIBB_GENERAL_DATA`. Implement a call to the parent class. Save and activate the redefinition.
- Set an external breakpoint in the redefined method of class `ZCL_BS_CU_GUIBB_GENERAL_DATA`.
- Start the user interface for MDG-C.
- Navigate to the edit page of *ERP Customer* data. Ensure that the UIBB *ERP Customer: Additional Data* is visible. The debugger shall stop in your custom class.

3.4 Create a custom genIL implementation class

For some use cases it might be required to create a custom genIL enhancement, too. Start first with creating a corresponding genIL implementation class.

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

- Name the new class `ZCL_BS_GENIL_CUSTOM`.
- Maintain the class description as *My custom genIL implementation*.
- Define the class as *Usual ABAP Class*.
- Decide if the class is *Final* or not.

Switch to the *Properties* tab and add the `CL_BS_CUSTOMER_GENIL` superclass. This is strictly required to ensure that existing SAP owned code is still being executed correctly. Choosing a different class will result in undesired and invalid side-effects. Remember the strict inheritance of SAP defined genIL classes:

- `CL_MDG_BS_GENIL`: the generic genIL implementation for all MDG solutions
    - `CL_BS_GENIL_BUPA`: the genIL implementation for MDG-BP

- `CL_BS_GENIL_MLT_ASSIGNMENTS`: the genIL implementation for multiple assignments
    - `CL_BS_GENIL_SUPPLIER`: the genIL implementation for MDG-S
        - `CL_BS_GENIL_CUSTOMER`: the genIL implementation for all MDG-C

Save, check, and activate 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.

3.5 Create a custom genIL enhancement

For the purpose of MDG-BP, MDG-C and MDG-S, SAP delivers the genIL component `BUPA` and an enhancement for this component called `BUPA_CUSP`. You can review both in transaction `GENIL_MODEL_BROWSER`.

If you want to enhance the genIL implementation, it is strictly required to create an enhancement of the existing enhancement `BUPA_CUSP`. Choosing a different enhancement, or the component `BUPA`, will result in undesired and invalid side-effects.

Run transaction `GENIL_MODEL_BROWSER`. Define your enhancement as `Z_CUSTOM_GENIL` and push the *Create* button.

In the following pop-up, enter a meaningful description (e.g., *My custom genIL Enhancement for MDG-BP/C/S*). Select the checkbox *Superenhancement* and maintain its value as `BUPA_CUSP`. A prefix is not required. Decide whether your enhancement is final, or not.

In the next screen, maintain your previously created genIL implementation class `ZCL_BS_GENIL_CUSTOM`. Save your enhancement.

3.6 Add custom logic

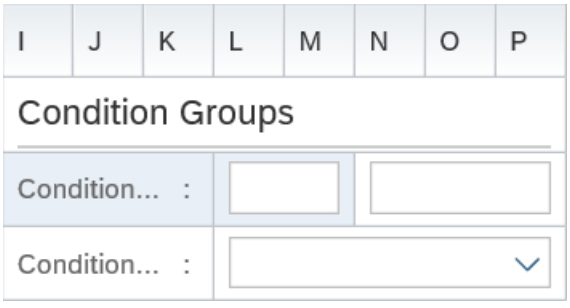
3.6.1 Replace drop-down list boxes with input fields using a search help

A few attributes are using drop-down list boxes in the SAP delivered user interfaces. This is the case if:

- The attribute values originate from domain fixed values.
- The expected number of values in a customizing table is low.

If your customizing table contains many entries, the usage of a drop-down list box can be complicated due to the required scrolling of the list. The following steps describe how to replace the drop-down list box with an input and text field including a search help. The example is a field of the UIBB *BS_CU_GEN_ADDITIONAL – ERP Customer: Additional Data*.

Go to the customizing that you have created for UIBB configuration *BS_CU_GEN_ADDITIONAL – ERP Customer: Additional Data*. There are five *Condition Group* fields that use the drop-down list box as SAP defined display type. Choose field *Condition Group 1 (KDKG1)* and change its display type to *Input Field*. Instead of the drop-down list box the UIBB shows an input field for users being able to maintain or search the key value. If you want to display the text, too, add field *KDKG1__TEXT* as *Input*



Field or *Text View* without label next to the *Condition Group 1 (KDKG1)* input field. Save all changes.

Call the user interface for MDG-C. Create a new or change an existing customer. Go to the *ERP Customer* edit page to view the *ERP Customer: Additional Data* UIBB. If you are in edit mode, you will notice that the field is displayed as an input field with a search help. The search help and the text display work out-of-the-box. This due to a generic search help implementation which uses the existing value set.

3.6.2 Search helps vs. performance

A different side-effect of large customizing tables is the runtime required for determining the value sets usable for generic F4 helps. If you encounter performance issues when displaying the user interface, follow the instructions of SAP note 2105467 MDG Performance to collect some SAT measurements.

Performance issues related to search helps based upon value sets can be identified by:

- An entry for class `CL_USMD_GENERIC_GENIL_TEXT`, e.g., for method `GET_TEXT_VALUES_FROM_TEXT_TAB`
- An entry for the genIL implementation classes (`CL_BS_GENIL_BUPA`, `CL_BS_GENIL_SUPPLIER` or `CL_BS_GENIL_CUSTOMER`) and methods related to text determination (e.g., `IF_GENIL_APPL_INTLAY~GET_TEXT` or one of the methods being called in this method)

If these classes and methods show a high gross and/or net runtime, consider deactivating the automated determination of the value sets. To keep a search help for the end-user, implement a custom OVS search help to determine the values only on demand.

The following steps describe how to achieve this for the field *Condition Group 1 (KDKG1)* that you have used in the previous chapter.

Determine whether the field in question uses

- the generic text assistance class `CL_USMD_GENERIC_GENIL_TEXT`
- or a specific implementation in one of the genIL implementation classes.

To do so, start with method `IF_GENIL_APPL_INTLAY~GET_TEXT` in `CL_BS_GENIL_CUSTOMER`. Browse through the class hierarchy and check if you can find a dedicated method call, or an execution of the parent call. If the parent call handles the field, the generic text assistance class is used. This is the case for field *Condition Group 1 (KDKG1)*.

If you find a dedicated method for text determination for the field in question, create a redefinition of this method in your custom genIL implementation class. Return the method without texts.

To restrict the usage of the generic text assistance class for a specific field, call transaction `SM30` for table/view `USMD_TXT_FLD_EXC`. Enter data model BP, the name of the attribute structure and the corresponding field name. Find this information in transaction `GENIL_MODEL_BROWSER` by displaying the attribute structure of the genIL object used by the UIBB. The correct values for the field *Condition Group 1 (KDKG1)* are:

- genIL (Dependent) Object as defined by the UIBB = *CU_GeneralData*
- Attribute Structure Name = *BSS_CUIL_GENERAL*
- Field Name = *KDKG1*

Save your changes. Run report `USMD_TXT_FLD_MAP_DELETE` for data model BP. This is required to update the change you maintained for excluding the field from the automated determination of the values. If you

start the UI for MDG-C, you will see that the field does not have any value help since its values set is not determined automatically anymore.

Call `SE24` or `SE80` for the custom feeder class `ZCL_BS_CU_GUIBB_GENERAL_DATA`.

Redefine method `IF_FPM_GUIBB_FORM~GET_DEFINITION`. This method is used to define the feeder class/UIBB. Use it to set the OVS search help for the *Condition Group 1 (KDKG1)* field to the current class by changing the corresponding value in `ET_FIELD_DESCRIPTION`.

Redefine method `OVS_HANDLE_PHASE_2`. Phase 2 of the OVS processing is responsible for determining the values to be displayed in the search help pop-up. To learn more about OVS in general, read the corresponding chapters in the FPM cookbook.

The method is called for each field of the current feeder class which uses an OVS. It's advisable to implement a `CASE` statement for the fieldnames. The `OTHERS` branch calls the parent class. The branch for the field in question implements the determination of the values to be displayed in the search help pop-up

- A simple select reading all values from the corresponding customizing table
- A more complex logic to accept user input before searching
- …

Review the existing OVS implementations of the SAP delivered feeder classes for examples. The appendix contains a simple select for displaying all values as example code.

Redefine methods `GET_ENTITY_DATA` respectively `GET_JOIN_ENTITY_DATA`. The methods are responsible to prepare the display of the UIBB fields. A call to the parent implementation ensures that all key values are read from the MDG Framework. Additional text values can be determined within the methods' code.

In our example, we need to reread the text for field *Condition Group 1 (KDKG1)*. The generic code does not work anymore since there is no value set which contains the text of the selected key. A simple select of the current text from the customizing table is required. The code needs to be implemented in method `GET_JOIN_ENTITY_DATA` since the field belongs to a joined genIL object (see the feeder class parameters in the UIBB configuration). The appendix contains the relevant code.

3.6.3 Field properties

Feeder classes of both form and list UIBBs contain the method `GET_FIELD_UI_PROP`. When using this method, be aware that it changes the field properties on the UI layer only.

If a field is specified as *hidden* or *read-only* in the MDG Framework or in the genIL layer, a corresponding call to the parent class will return the field property as *read-only*. Your custom code can surely change this setting, but it will not have the desired effect. Since you change the property only in the UI layer, but not in genIL, any entered value will be deleted by the genIL layer. There, the field is still *hidden* or *read-only*, which means that value changes are not allowed.

3.6.4 Copying SAP delivered code

Prevent using any (deep) copy of SAP delivered overview pages and / or UIBB configurations including feeder classes. Copied objects cannot be improved or corrected by SAP notes. Any SAP note you apply in your system for improvements or bug fixes must be applied manually to copied objects. Otherwise, your custom objects and will run outdated and invalid code, likely causing undesired side-effects.

4 Performance

FPM allows multiple options for enhancing existing user interfaces. Nevertheless, it is strongly recommended to use customizing only. At runtime, the Web Dynpro rendering needs to determine the *final* page to be displayed. The more options you use, the more runtime is required to determine the *final* page:

- Do not use customizing and enhancements in parallel.
- Use context-based adaptations only if you really require a dynamic change of the page's layout.
- Consider whether you want to allow end-user personalization or not.

5 Appendix

Source code of the mentioned classes and methods.

5.1 ZCL_BS_CU_GUIBB_GENERAL_DATA

5.1.1 IF_FPM_GUIBB_FORM~GET_DEFINITION

```
  METHOD if_fpm_guibb_form~get_definition.
*! Get the definition of the feeder class / UIBB.
    "inherit first
    super->if_fpm_guibb_form~get_definition(
      IMPORTING
        eo_field_catalog         = eo_field_catalog
        et_field_description     = et_field_description
        et_action_definition     = et_action_definition
        et_special_groups        = et_special_groups
        et_dnd_definition        = et_dnd_definition
        es_options               = es_options
        es_message               = es_message
        ev_additional_error_info = ev_additional_error_info ).

    "set the OVS search help for field Condition Group 1 (KDKG1) to the current
class
    READ TABLE et_field_description ASSIGNING FIELD-SYMBOL(<description>)
      WITH KEY name = 'KDKG1'.
    IF sy-subrc = 0.
      <description>-ovs_name = me->class_name.
    ENDIF.
  ENDMETHOD.
```

5.1.2 OVS_HANDLE_PHASE_2

```
  METHOD ovs_handle_phase_2.
    CASE iv_field_name.
      WHEN 'KDKG1'.
        "Implement the OVS search result for the Condition Group 1.
        "The corresponding customizing tables are TVKGG & TVKGGT.
        TYPES:
          BEGIN OF ty_s_ovs_condition,
            kdkg1       TYPE kdkgr,
            kdkg1__text TYPE vtext,
          END OF  ty_s_ovs_condition,
          ty_t_ovs_condition TYPE STANDARD TABLE OF ty_s_ovs_condition.
        CREATE DATA er_output TYPE ty_t_ovs_condition.
        FIELD-SYMBOLS <output> TYPE  ty_t_ovs_condition.
        ASSIGN er_output->* TO <output>.
        SELECT * FROM tvkgg INTO TABLE @DATA(conditions).
        IF conditions IS NOT INITIAL.
          SELECT * FROM tvkggt INTO TABLE @DATA(texts)
            FOR ALL ENTRIES IN @conditions
            WHERE kdkgr = @conditions-kdkgr
              AND spras = @sy-langu.
        ENDIF.
```

```
        LOOP AT conditions ASSIGNING FIELD-SYMBOL(<condition>).
          APPEND INITIAL LINE TO <output> ASSIGNING FIELD-SYMBOL(<line>).
          <line>-kdkg1 = <condition>-kdkgr.
          <line>-kdkg1__text = <condition>-kdkgr.
          READ TABLE texts ASSIGNING FIELD-SYMBOL(<text>)
            WITH KEY kdkgr = <condition>-kdkgr
                     spras = sy-langu.
          CHECK sy-subrc = 0
            AND <text>-vtext IS NOT INITIAL.
          <line>-kdkg1__text = <text>-vtext.
        ENDLOOP.

      WHEN OTHERS.
        "The method is called for each field using an OVS. To support
        "SAP owned fields with their standard logic, the OTHERS simply
        "calls the parent class.
        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.
  ENDMETHOD.
```

5.1.3 GET_JOIN_ENTITY_DATA

```
  METHOD get_join_entity_data.
*! Get data for the joined entities (genIL objects) of the UIBB.
    "inherit first
    super->get_join_entity_data(
      EXPORTING
        io_entity = io_entity
        is_join   = is_join
      CHANGING
        cs_data   = cs_data ).

    IF my_active <> abap_true.
      RETURN.
    ENDIF.

    "supply the text for field Condition Group 1 (KDKG1)
    ASSIGN COMPONENT 'KDKG1__TEXT' OF STRUCTURE cs_data TO FIELD-SYMBOL(<text>).
    IF sy-subrc = 0.
      ASSIGN COMPONENT 'KDKG1' OF STRUCTURE cs_data TO FIELD-SYMBOL(<key>).
      IF sy-subrc = 0
        AND <key> IS NOT INITIAL.
        SELECT SINGLE * FROM tvkggt INTO @DATA(text)
          WHERE kdkgr = @<key>
            AND spras = @sy-langu.
        IF TEXT-vtext IS NOT INITIAL.
          <text> = TEXT-vtext.
```

```
        ELSE.
          <text> = <key>.
        ENDIF.
      ENDIF.
    ENDIF.
  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
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
3194967	MDG Customer Connection 2021 for S/4HANA 2022
3043582	MDG Customer Connection 2020
2479869	Usage of Lean Classification with SAP Master Data Governance

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)