PUBLIC

How To: Extend MDG-F Data Model 0G by New Fields

Applicable Releases:
All

Version 1.5

October 2022

Document History

Document Version	Description
1.5	Minor textual updates and updated links
1.4	Updated Layout
1.3	Updated links for new SAP Community
1.2	Additional information for data transfer
1.1	Updated broken links
1.0	First official release of this guide

THE BEST RUN

1. BUSINESS SCENARIO

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

This How to Guide describes the solution to extend an existing entity type with storage and use type 1 with some new fields. It covers the key concepts and implementation details in general and includes a real-life example of the MDG-F data model 0G.

2. SAP NOTES AND LINKS

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

1637249 MDG: Information for efficient message processing

2021246 MDG Financials – (Un-) Supported Segments and Fields

2105467 MDG Performance

2337685 MDG-F: Standard Validations of Data Model 0G

Configuration and Enhancement of SAP Master Data Governance

Web Dynpro & Floorplan Manager

3. BACKGROUND INFORMATION

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)

In SAP ECC systems a Cost Center can be used for Joint Venture Accounting. Joint venture accounting has been an add-on to SAP ECC before SAP ECC 6.0. Starting with SAP ECC 6.0 it is a built-in feature that needs activation via business function(s). It adds some field to the SAP ECC cost center master data:

- Joint Venture
- Recovery Indicator
- Equity Type
- Joint Venture Object Type
- JIB/JIBE Class & Subclass A

The fields cannot be pre-defined by SAP within MDG-F data model 0G. They are valid only if both the MDG-F and Joint Venture Accounting business functions are active.

In S/4 HANA, from release 1610 on, the business function for Joint Venture Accounting is already active (Always-On business function).

The scenario assumes that you have activated Joint Venture Accounting in your system and now want to add the fields to MDG-F, too.

4. STEP BY STEP EXPLANATION

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. Preparation of the extension by collecting all required information about the fields to be added from the source system.
   a. Analyze the field attributes in the source system. Ask the following questions:
      i. Which data element is used?
      ii. Does the field use a check table?
      iii. If yes, how is the table structured (single or multiple key fields)?
   b. Translate the field attributes into the MDG-F specific format. Decide whether the field be modeled as an attribute or whether a relation (e.g. entity type with SU Type 3) needed.
   c. Identify further development items according to the source system. Ask the following questions:
      i. How are the fields represented in the backend user interface?
      ii. Are the fields already part of data replication technologies or is an extension required?
2. Extension of the MDG-F data model 0G.
   a. Add attributes and/or relations.
   b. Activate the data model and generate the model-specific structures.
3. Extension of the User Interface.
   a. Add new fields to an existing UIBB or a new UIBB (optional).
4. Validation (optional).
   a. Implement new checks in USMD_RULE_SERVICE.
5. Replication (optional).
   a. Replicate using ALE IDocs or SOA Services.

4.1. Preparation of the Extension

The scenario is adding the Joint Venture Accounting fields of the SAP ECC Cost Center to the MDG data model 0G. The following steps are recommended to prepare the actual work. The information that is collected during the preparation suites as the foundation for the extension.

4.1.1. Analyze the Field Attributes in the Source System

The fields Joint Venture, Recovery Indicator, Equity Type, Joint Venture Object Type, and JIB/JIBE classes belong to the cost center object that is stored in data base table CSKS. Using the transaction SE11, the following information can be retrieved about the fields:

Field	Name	Type	Check Table	Comment
Joint Venture	VNAME	JV_NAME CHAR (6)	T8JV	Check table uses the company code as foreign key.
Recovery Indicator	RECID	JV_RECIND CHAR (2)	T8JJ	Check table uses the company code as foreign key.
Equity Type	ETYPE	JV_ETYPE CHAR (3)	T8JG	Check tables uses the company code and joint venture as foreign keys. Further key component is a valid from date
Joint Venture Object Type	JV_OTYPE	JV_OTYPE CHAR (4)	-	-
JIB/JIBE Class	JV_JIBCL	JV_JIBCL CHAR (3)	T8J6A	Check table uses the company code as foreign key.
JIB/JIBE Subclass A	JV_JIBSA	JV_JIBSA CHAR (4)	T8J6C	Check table uses the company code and the JIB/JIBE Class as foreign keys.

4.1.2. Translate the Field Attributes into the MDG-F Specific Format

Fields Joint Venture, Recovery Indicator, Equity Type and the JIB/JIBE classes all have a check table using the company code and, in some cases, further fields as foreign keys. You cannot model these fields as simple attributes within the MDG data model. You must use entity type with SU Type 3 and relations instead.

Field Joint Venture Object Type has no check table. You can model this field as a simple attribute within the MDG data model.

4.1.3. Identify Further Development Items According to the Source System

Joint Venture fields are displayed in a specific tab during the maintenance of the cost center master data in SAP ECC. The same is possible in the MDG-F user interface by using a custom form UIBB.

Decide whether the fields are relevant for data transfer scenarios, or not.

4.2. Extension of the MDG-F data model 0G

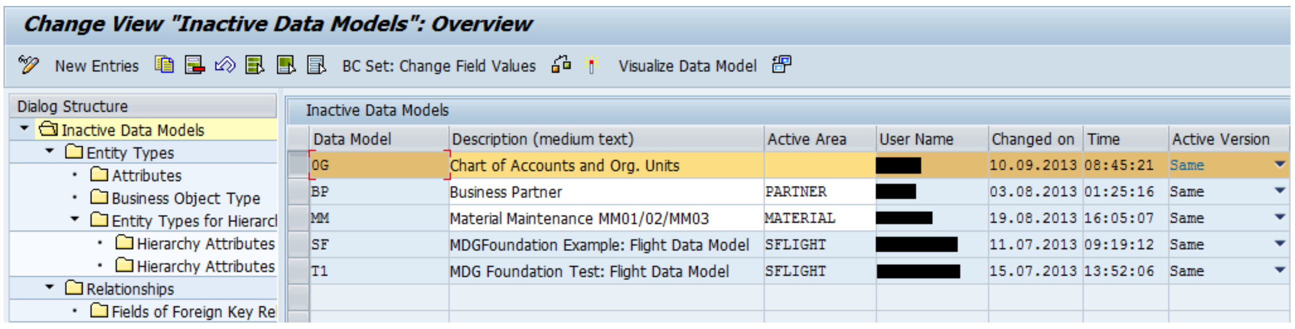
The scenario is adding the Joint Venture Accounting fields of the SAP ECC Cost Center to the MDG data model 0G. The following steps add the required fields to the data model. They use the information collected by the previous preparation step.

4.2.1. Prepare the data model enhancement

Adding new fields to the MDG-F data model 0G is realized as an enhancement of the data model. Applying SAP bug fixes and patches does not invalidate your enhancements.

1. Logon to your MDG hub system.
2. Start transaction MDGIMG.
3. If you want to transport your enhancements, you must assign a package for your customizing includes.
   a. In Customizing for Master Data Governance (transaction MDGIMG), open the activity General Settings → Data Modeling → Assign Package for Customizing Include.
   b. Create a new entry for Data Model 0G and the desired package to be used.
   c. Save your entry.
4. In Customizing for Master Data Governance (transaction MDGIMG), open the activity General Settings → Data Modeling → Edit Data Model.

You are now using the MDG data modeling workbench. Each of the following steps must be performed using the workbench for data model 0G.



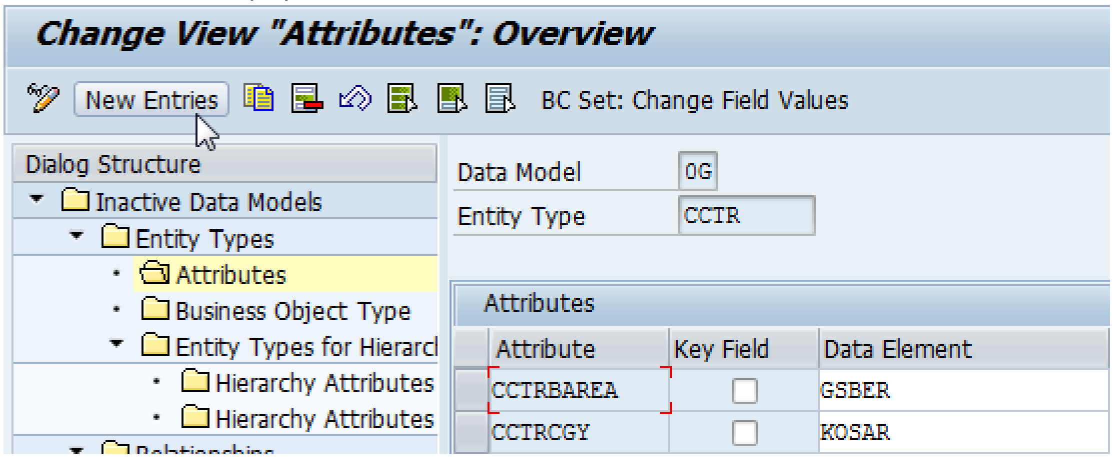
Data Model	Description (medium text)	Active Area	User Name	Changed on	Time	Active Version
0G	Chart of Accounts and Org. Units		[redacted]	10.09.2013	08:45:21	Same
BP	Business Partner	PARTNER	[redacted]	03.08.2013	01:25:16	Same
MM	Material Maintenance MM01/02/MM03	MATERIAL	[redacted]	19.08.2013	16:05:07	Same
SF	MDGFoundation Example: Flight Data Model	SFLIGHT	[redacted]	11.07.2013	09:19:12	Same
T1	MDG Foundation Test: Flight Data Model	SFLIGHT	[redacted]	15.07.2013	13:52:06	Same

4.2.2. Add fields as attributes

Attributes of an entity type with SU Type 1 are "simple" fields. These fields either have no check table at all (for example plain text, dates, and so on), or have a check table that consists of a single key field only (without foreign keys or dependencies to other values).

Add field Joint Venture Object Type as a simple attribute to the Cost Center entity type in the MDG data model.

1. In the MDG data modeling workbench mark data model 0G and double click Entity Types in the tree menu on the left.
2. Locate and select entity type CCTR (Cost Center).
3. Double click Attributes in the tree menu on the left.
4. Click the New Entries (F5) button.



Data Model: 0G
Entity Type: CCTR

Attributes

Attribute	Key Field	Data Element
CCTRBAREA	☐	GSBER
CCTRCGY	☐	KOSAR

5. Define Attribute ZJV_OTYPE using Data Element JV_OTYPE.
6. Save (CRTL+S) your enhancement.

You have successfully added the new field Joint Venture Object Type as a simple attribute to the Cost Center in data model 0G.

4.2.3. Add fields using entity types with SU Type 3 and relations

Entity types with SU Type 3 are usable for “complex” fields. These fields have a check table that consists of multiple key fields using foreign keys. The actual key depends on other fields (which might have to be added as entity types with SU Type 3 to the data model, too). Relations are required to link the entities with each other.

Add fields Joint Venture, Recovery Indicator, Equity Type and the JIB/JIBE classes as entity types with SU Type 3. Use relation to link the entities with their foreign keys as well as the cost center.

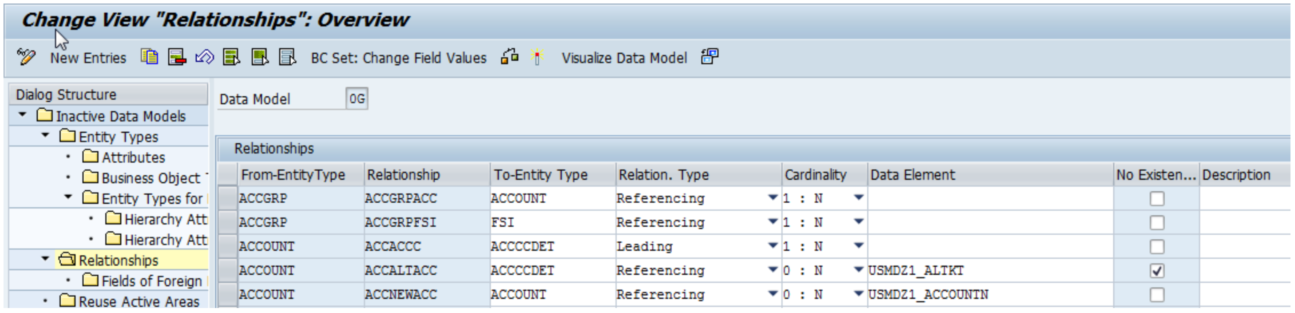
1. In the MDG data modeling workbench mark data model 0G and double click Entity Types in the tree menu on the left.
2. Choose button New Entries (F5) to create new entity types for each field to be added. Define the following values for field Joint Venture:

Attribute	Value	Explanation
Entity Type	ZVNAME	Re-use the field’s name. If you add an SAP field, use the customer namespace.
Storage / Use Type	Not Changeable via MDG; No Generated Tables	This defines a entity types with SU Type 3 type that is used for foreign key relations in check tables.
Data Element	JV_NAME	Re-use the data element of the field to be added.
Hierarchies	No Hierarchy	Not supported for entity types with SU Type 3.
Validity / Hierarchy	Hierarchy is not Edition Dependent	Not supported for entity types with SU Type 3.
Key Assignment	Key Cannot Be Changed; No Internal Key Assignment	Not supported for entity types with SU Type 3.
Description	Joint Venture	Re-use the field’s description
Others	Space	It is not recommended to maintain other attributes for Entity types with SU Type 3.

3. Repeat step 2 for fields:

  a. Recovery Indicator
  b. Equity Type
  c. JIB/JIBE Class
  d. JIB/JIBE Subclass A

4. You have added the new fields as entity types with SU Type 3.
5. Optional: relations for foreign keys require that the actual foreign key field exists in the data model.
   a. Check if all foreign key fields exist.
   b. If fields are missing, add the same as entity types with SU Type 3 by repeating step 2 for the foreign key.
   c. The Joint Venture Accounting scenario uses the company code as a foreign key. The company code is already part of the data model 0G. It is represented by the COMPCODE entity type (SU Type 3).
6. Switch to the Relationships view of the MDG data modeling workbench.



Data Model 0G

Relationships

From-EntityType	Relationship	To-Entity Type	Relation. Type	Cardinality	Data Element	No Existen...	Description
ACCGRP	ACCGRPACC	ACCOUNT	Referencing	1 : N		☐	
ACCGRP	ACCGRPFSI	FSI	Referencing	1 : N		☐	
ACCOUNT	ACCACCC	ACCCCDET	Leading	1 : N		☐	
ACCOUNT	ACCALTACC	ACCCCDET	Referencing	0 : N	USMDZ1_ALTKT	✓	
ACCOUNT	ACCNEWACC	ACCOUNT	Referencing	0 : N	USMDZ1_ACCOUNTN	☐	

7. Choose the New Entries (F5) button to create new relations for foreign keys. Define the following values for the relation from the Company Code to the Joint Venture:

Attribute	Value	Explanation
From-Entity Type	COMPCODE	Defines the entity type for the foreign key field.
Relationship	ZCCVNAME	Define a name for your relationship (CHAR 9)
To-Entity Type	ZVNAME	Define the entity type with SU Type 3 representing your new field.
Relation. Type	Leading	The foreing key “leads” the relation to your new field.
Cardinality	1 : N	The same foreign key may “lead” multiple relations (e.g. company code 0001 can lead multiple joint ventures).
No Existence Check	Space	The generic existence check provided by the MDG framework is mandatory for leading relations
Description	Company Codes for Joint Ventures	You can maintain any description.
Others	Space	It is not recommended to maintain other attributes for relations.

8. Repeat step 7 for the following relations:
   a. Company Code to Recovery Indicator
   b. Company Code to Equity Type
   c. Company Code to JIB/JIBE Class
   d. Company Code to JIB/JIBE Subclass A
9. Field JIB/JIBE Subclass A uses JIB/JIBE Class as foreign key, too. Therefore, you need to create one more relation by repeating step 7 for JIB/JIBE Class to JIB/JIBE Subclass A.
10. You have defined the relations needed for the foreign keys.
11. Now add the relations of the new fields to the entity type with SU Type 1 (CCTR).
12. Choose button New Entries (F5) to create new relations for the new fields to the entity type with SU Type 1. Define the following values for the relation from the Joint Venture to the Cost Center:

Attribute	Value	Explanation
From-Entity Type	ZVNAME	Defines the entity type for the new field.

THE BEST RUN SAP

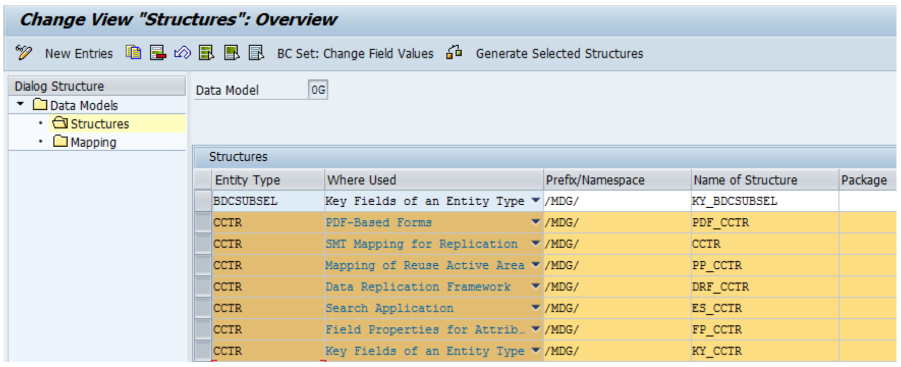
Attribute	Value	Explanation
Relationship	ZCTRVNAME	Define a name for your relationship (CHAR 9)
To-Entity Type	CCTR	Define the entity type with SU Type 1 that shall user your new field.
Relation. Type	Referencing	Your new field is used as a reference by the entity type with SU Type 1.
Cardinality	0 : N	It’s optional to enter a value for the new field.
No Existence Check	True	The generic existence check provided by the MDG framework could be used by referencing relations. Errors would be indicated by a generic message. The how-to guide implements specific validations and thus does not need the generic check.
Description	Joint Ventures for Cost Centers	You can maintain any description.
Others	Space	It is not recommended to maintain other attributes for relations.

13. Repeat step 12 for the relations:
    a. Recovery Indicator to Cost Center
    b. Equity Type to Cost Center
    c. JIB/JIBE Class to Cost Center
    d. JIB/JIBE Subclass A to Cost Center
14. Save (CRTL+S) your enhancement.
15. You have added all fields for the Joint Venture Accounting of cost centers to the data model.

4.2.4. Activate the data model and generate the model-specific structures

As soon as you have added all fields to the data model, you must activate the same. The activation of the data model usually triggers the re-generation of the model-specific structures automatically. Nevertheless, the following steps describe what needs to be done if the automated re-generation does not happen.

1. Switch back to the Inactive Data Models view of the MDG data modeling workbench.
2. Mark data model 0G and choose the Check (Ctrl+Shift+F1) button. The activation of the data model only works if there are no error messages.
3. Choose the Activate (Ctrl+Shift+F3) button. The activation may take some time.
4. If the activation issues the message “Change requests must be adjusted because data model 0G has been modified (USMD1B040)”, confirm the message choose the button Adjust Staging Area of Linked Change Requests (Ctrl+F12).
5. If a pop-up “Possible loss of Data – Was the mapping between the reuse active area and the data model adjusted?” is issued by the system, confirm by clicking Yes.
6. You have successfully enhanced the data model 0G.
7. To check if everything worked fine, the following steps are recommended.
8. Choose button Visualize Data Model (Ctrl+Shift+F6).
9. Choose button Active Version (Ctrl+F2).
10. Locate your enhanced entity type (CCTR) and check that your fields are visible.
11. Start transaction SE80 for package /MDG/FIN_MDGF_STRUCTURES.
12. Locate the generated structures for your enhanced entity type, e.g. structure /MDG/_S_0G_PP_CCTR named Source Structure for PP Mapping of the Cost Center.
13. Check that your fields are visible within the custom include of the structure. The custom include’s name starts with CI_MDG_S_*.
14. If your fields are not visible within the custom include, re-generate the structures.
15. Start transaction MDGIMG.
16. In the MDGIMG menu tree navigate to General Settings → Data Modeling → Generate Data Model-Specific Structures.
17. Mark data model 0G and double click Structures in the tree menu on the left.


Change View "Structures": Overview
New Entries
BC Set: Change Field Values
Generate Selected Structures
Dialog Structure
Data Models
Structures
Mapping

Data Model 0G

Structures

Entity Type	Where Used	Prefix/Namespace	Name of Structure	Package
BDCSUBSEL	Key Fields of an Entity Type	/MDG/	KY_BDCSUBSEL	
CCTR	PDF-Based Forms	/MDG/	PDF_CCTR	
CCTR	SMT Mapping for Replication	/MDG/	CCTR	
CCTR	Mapping of Reuse Active Area	/MDG/	PP_CCTR	
CCTR	Data Replication Framework	/MDG/	DRF_CCTR	
CCTR	Search Application	/MDG/	ES_CCTR	
CCTR	Field Properties for Attrib…	/MDG/	FP_CCTR	
CCTR	Key Fields of an Entity Type	/MDG/	KY_CCTR	

18. Mark all entries of your enhanced entity type (in our scenario CCTR) and choose button Generate Selected Structures (Ctrl+Shift+F3). The structure generation may take some time.
19. Repeat the visibility check as described in steps 8 to 10.
20. You have re-generated the data mode-specific structures.

4.3. Extension of the User Interface

The scenario is adding the Joint Venture Accounting fields of the SAP ECC Cost Center to the MDG data model 0G. The following steps add the required fields to the user interface. The fields use the generated structures created when you extended the MDG-F data model 0G.

4.3.1. Upgrade and/or migration of “old” user interface enhancements

If you already use MDG-F with older MDG releases and you have enhanced your data model 0G, MDG-F 7.0 requires re-implementing your user interface enhancements due to a change of the user interface technology.

Adjustments of the data model are not required. Your enhancements of the data model are still valid and re-used within the new user interfaces. Nevertheless, it is mandatory that the data model-specific structures are generated at least once within MDG-F 7.0 (usually this has already been done as a recommended step during the general upgrade process from older MDG-F releases to MDG-F 7.0).

4.3.2. Adding new fields to an existing UIBB vs. creating a new UIBB for the new fields

You can either add new fields to the user interface either by adding them to existing SAP-defined UIBBs or to new custom UIBBs. There is no general recommendation for the one or the other possibility. The required effort is comparable. The following questions might help to find a decision:

- How many fields are you adding? If you are only adding one or two fields, it is easier to add them to an existing
- Do the fields require a custom feeder class? Such a feeder class may be necessary for complex search helps or specific field handling. If you need a custom feeder class, it could make sense to use a custom UIBB too. However, it is also possible to maintain a custom feeder class for an SAP-defined UIBB.
- How do you want the final user interface to look?

The how-to guide describes the required steps for both approaches. In your implementation, choose only one. Adding the fields twice to the same user interface does not make any sense. SAP development would use a custom, specific UIBB for the Joint Venture scenario.

4.3.3. Create a custom feeder class

Custom feeder classes are suitable for adding new or changing existing functionality. It is possible to implement a custom feeder class to change the pre-defined behavior of the SAP user interfaces. Nevertheless, the most common scenario for a custom feeder class is indeed the enhancement of the user interface with new fields that require some specific logic (e.g. complex OVS search helps).

- If a new field is a simple text field a custom feeder class is not required.
- If a new field uses a check table with a single key field only, a custom feeder class is not required.

- Following the Joint Venture Accounting scenario field Joint Venture Object Type does not require a custom feeder class. It can re-use the existing functionality of the cost center feeder class CL_MDGF_GUIBB_CCTR. Since the other fields require a complex search help due to the dependency on the company code, the implementation of a custom feeder class is required.

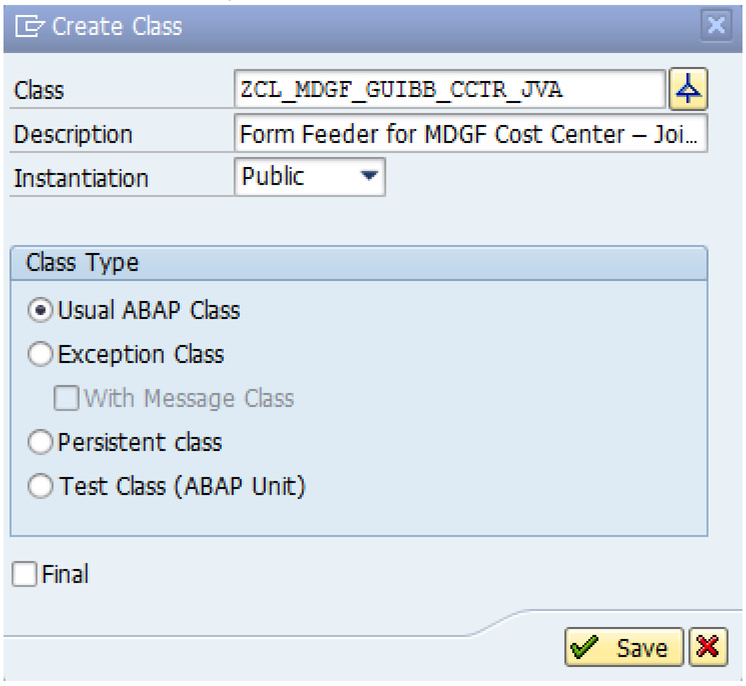
To simplify the implementation of custom feeder classes, SAP provides some generic feeder classes for MDG-F. You should make yourself familiar with their supported functionality before creating your own feeder.

- Generic Form Feeder for MDG-F Entities CL_MDGF_GUIBB_FORM
- Generic List Feeder for MDG-F Entities CL_MDGF_GUIBB_LIST
- Generic Search Result Feeder for MDGF Entities CL_MDGF_GUIBB_RESULT
- Further MDG-F entity specific feeder classes are available in package USMDZ10, too.

It is strongly recommended that your custom feeder class inherits from one of the mentioned classes depending on the related UIBBs type.

The following steps are required for the creation of a custom feeder class.

1. Start transaction SE80 and navigate to the package that shall contain the new class.
2. Create class ZCL_MDGF_GUIBB_CCTR_JVA using the description Form Feeder for MDGF Cost Center – Joint Venture Accounting. It's a usual ABAP class with public instantiation. It should not be final.

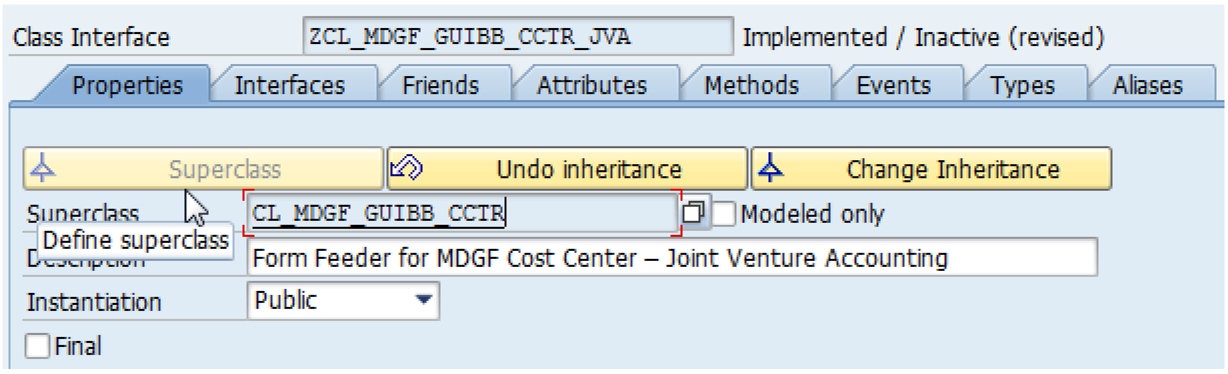


Class: ZCL_MDGF_GUIBB_CCTR_JVA
Description: Form Feeder for MDGF Cost Center – Joi...
Instantiation: Public

Class Type: [✓ Usual ABAP Class] Exception Class, With Message Class, Persistent class, Test Class (ABAP Unit)

Final: ☐

3. Switch to tab Properties and choose button Superclass to define the class inheritance.
   a. Since the fields must be included in a form UIBB, one option for inheritance is the generic form feeder class CL_MDGF_GUIBB_FORM.
   b. The Joint Venture Accounting scenario uses class CL_MDGF_GUIBB_CCTR instead! The cost center's feeder class already implements some logic that is re-usable for the Joint Venture Accounting, too (e.g. the determination of the currently selected company code and its storage in class attribute MV_COMPANY_CODE).


Class Interface ZCL_MDGF_GUIBB_CCTR_JVA Implemented / Inactive (revised)

Superclass CL_MDGF_GUIBB_CCTR Modeled only
Description Form Feeder for MDGF Cost Center – Joint Venture Accounting
Instantiation Public
Final

4. Activate the custom feeder class.

You have created a custom feeder class that is usable for a form UIBB. Re-definitions and or new methods are implemented in a later point in time.

4.3.4. Optional: Implement transient fields in the feeder class

Transient fields are fields that are not directly required for the master data governance process. The user interfaces include transient fields for displaying texts or descriptions belonging to a specific input field. This increases the usability of the complete user interface since displaying only key values without related texts might be difficult to understand for users.

The Joint Venture Account scenario needs transient field for the input fields Joint Venture, Recovery Indicator, Equity Type, and JIB/JIBE classes.

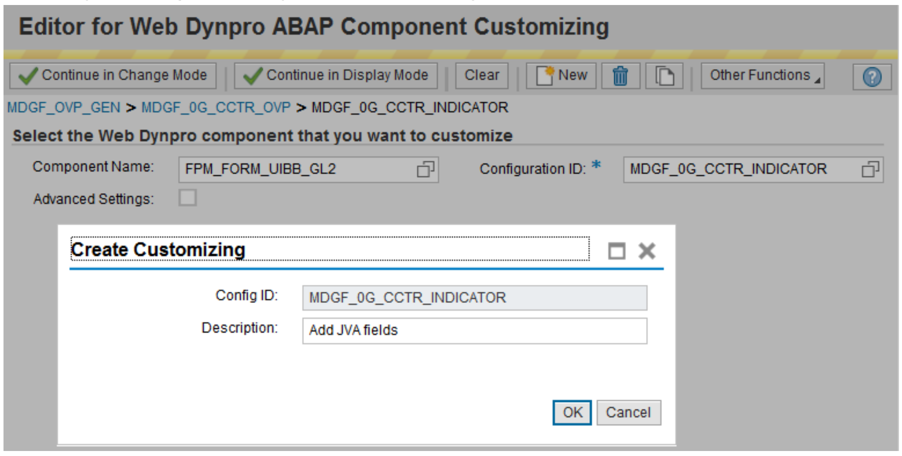
1. Redefine method CREATE_STRUCT_RTTI.
2. Implement a parent call to the super class first.
3. Implement transient fields. The fields' name shall be a concatenation of the related input fields' name and the suffix __TXT.
4. The source code is available in the appendix.

You have enhanced the field catalog with transient text fields.

4.3.5. Add fields to an existing UIBB (Option 1)

New fields can be added to an existing UIBB by customizing the same. The example adds the Joint Venture Accounting fields to the SAP form UIBB for Cost Center Indicators (MDGF_0G_CCTR_INDICATOR).

1. Start the web dynpro component customizing.
   a. Component Name = FPM_FORM_UIBB_GL2
   b. Configuration ID = MDGF_0G_CCTR_INDICATOR
2. Enter any description for your customizing and choose the OK button.


Editor for Web Dynpro ABAP Component Customizing

MDGF_OVP_GEN > MDGF_0G_CCTR_OVP > MDGF_0G_CCTR_INDICATOR

Select the Web Dynpro component that you want to customize

Component Name: FPM_FORM_UIBB_GL2 Configuration ID: * MDGF_0G_CCTR_INDICATOR
Advanced Settings:

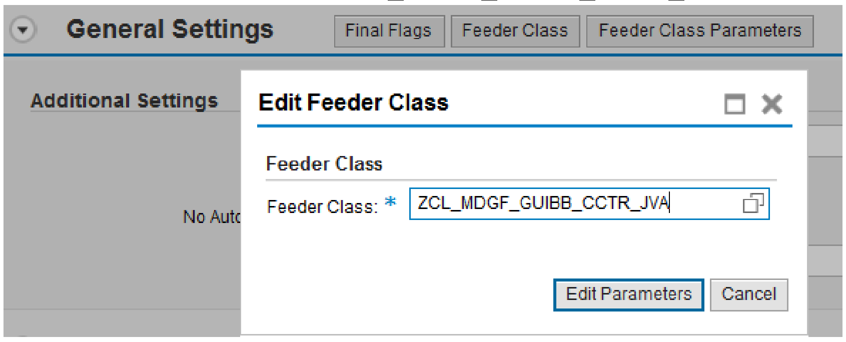
Create Customizing

Config ID: MDGF_0G_CCTR_INDICATOR
Description: Add JVA fields

3. Replace the pre-defined SAP feeder class with the previously created custom feeder class.

a. Choose button Feeder Class on the General Settings tray (you might have to expand the tray to see the button).

b. Enter Feeder Class ZCL_MDGF_GUIBB_CCTR_JVA and choose button Edit Parameters.



c. Confirm the pop-up Confirm changed feeder class with Yes. A data loss cannot happen since the custom feeder class inherits from the cost center feeder class that we have just replaced.

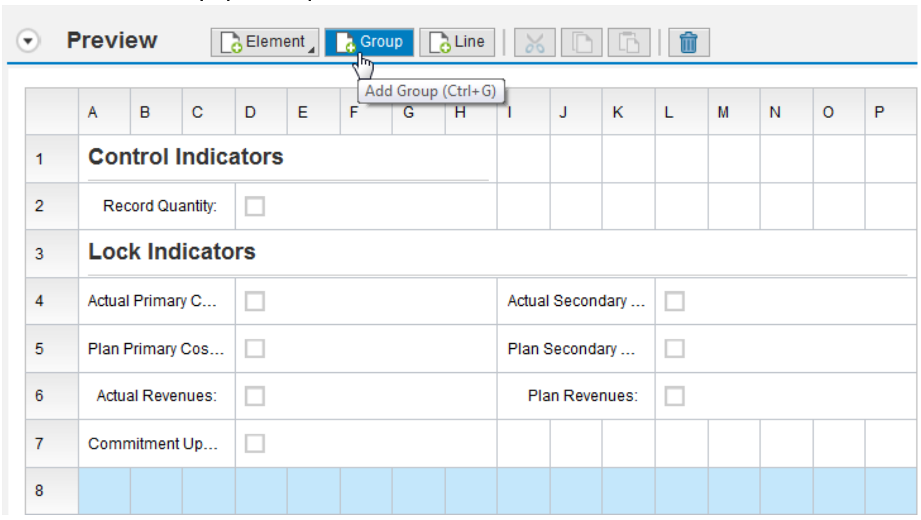
d. Confirm the pop-up Edit Parameters with OK. The editor re-uses the previously defined parameters. Changing them not required.

e. Save your changes.

4. Add a new group for the Joint Venture Accounting fields.

a. Groups can be added in various ways. It is possible to use the trays Preview and Form UIBB Schema.

i. On the Preview tray mark any empty line below the existing group for Lock Indicators choose button Add Group (Ctrl+G).


Preview
Element
Group
Line
Add Group (Ctrl+G)
A B C D E F G H I J K L M N O P
1 Control Indicators
2 Record Quantity:
3 Lock Indicators
4 Actual Primary C... Actual Secondary ...
5 Plan Primary Cos... Plan Secondary ...
6 Actual Revenues: Plan Revenues:
7 Commitment Up...
8

ii. On the Form UIBB Schema tray choose button Add Element and choose Add Group.

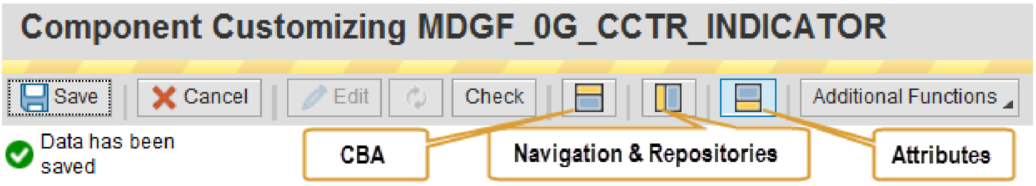
b. Use the Attributes of Group frame to define the Group Header as Joint Venture Accounting.

c. Save your changes.

5. Add the fields to the group.

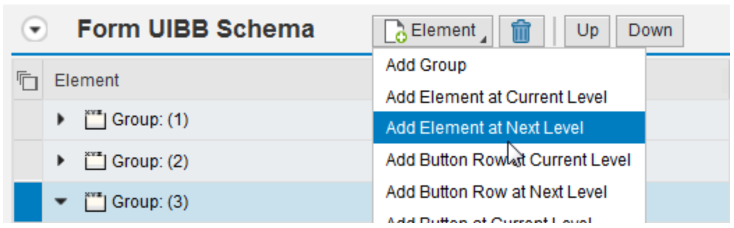
a. Fields can be added in various ways using the Form UIBB Schema.

i. Use Navigation & Repositories to select fields (you might have to enable the frame by choosing the related button in the top menu bar). Drag and drop selected fields into the group.



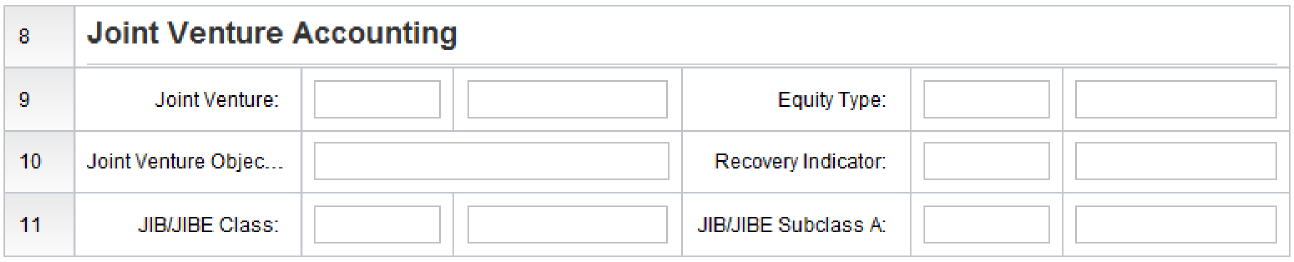
ii. On the Form UIBB Schema tray select the group, choose button Add Element and choose Add Element on Next Level. Select the fields to be added from the resulting pop-up.

THE BEST RUN SAP


Form UIBB Schema
Element
Up
Down
Element
Group: (1)
Group: (2)
Group: (3)
Add Group
Add Element at Current Level
Add Element at Next Level
Add Button Row at Current Level
Add Button Row at Next Level

b. Adjust the layout of the group as well as the field attributes.
   i. Selecting each field and use the Attributes of Element frame.
   ii. Ensure that common input fields always use FPM_REFRESH as FPM Event ID for onEnter.
c. Save your changes.

You have added the fields to your user interface. The screenshot below shows a possible result.

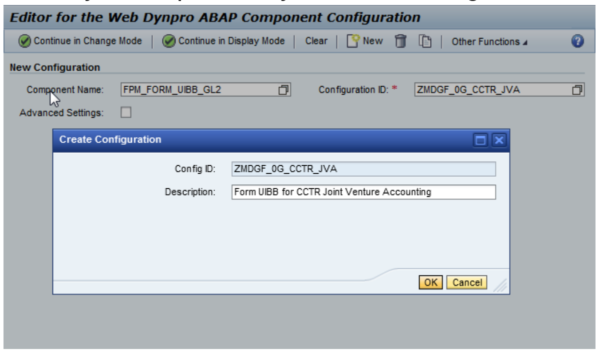


8	Joint Venture Accounting					
9	Joint Venture:			Equity Type:		
10	Joint Venture Objec...			Recovery Indicator:		
11	JIB/JIBE Class:			JIB/JIBE Subclass A:		

4.3.6. *Add fields as a new UIBB (Option 2)*

New fields can be added within a new UIBB. The example creates a new form UIBB for the Joint Venture Accounting fields and adds the same to the SAP OVP for Cost Centers.

1. Start the web dynpro component configuration.
   a. Component Name = FPM_FORM_UIBB_GL2
   b. Configuration ID = ZMDGF_0G_CCTR_JVA
2. Enter any description for your customizing and choose the OK button.



Editor for the Web Dynpro ABAP Component Configuration

Continue in Change Mode | Continue in Display Mode | Clear | New | Other Functions

New Configuration

Component Name: FPM_FORM_UIBB_GL2
Configuration ID: * ZMDGF_0G_CCTR_JVA
Advanced Settings: ☐

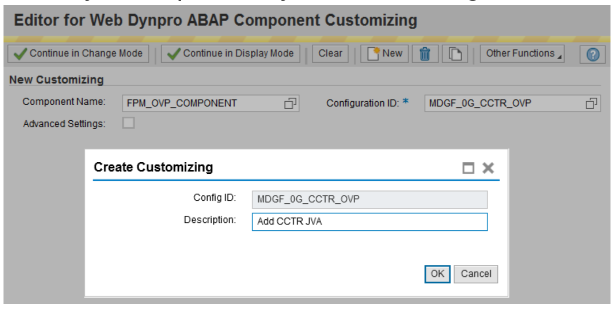
Create Configuration

Config ID: ZMDGF_0G_CCTR_JVA
Description: Form UIBB for CCTR Joint Venture Accounting

OK Cancel

3. Define the feeder class and its parameters.
   a. Enter Feeder Class ZCL_MDGF_GUIBB_CCTR_JVA and choose button Edit Parameters.
   b. Enter parameter Component as MDGF.
   c. Enter parameter Object Name as CCTR.
   d. Tick the checkbox for parameter Editable.
   e. Confirm the pop-up Edit Parameters with OK.
4. The configuration edition automatically creates a single group adding all fields that are defined for the object (cost center). Follow the steps 4 and 5 of “Add fields to an existing UIBB” to define the layout of your new UIBB.
5. Save your changes.
6. The new form UIBB is now completed. It can be added to the OVP by customizing.
7. Start the web dynpro component configuration.
   a. Component Name = FPM_OVP_COMPONENT
   b. Configuration ID = MDGF_0G_CCTR_OVP

8. Enter any description for your customizing and choose the OK button.



Editor for Web Dynpro ABAP Component Customizing

New Customizing

Component Name: FPM_OVP_COMPONENT

Configuration ID: * MDGF_0G_CCTR_OVP

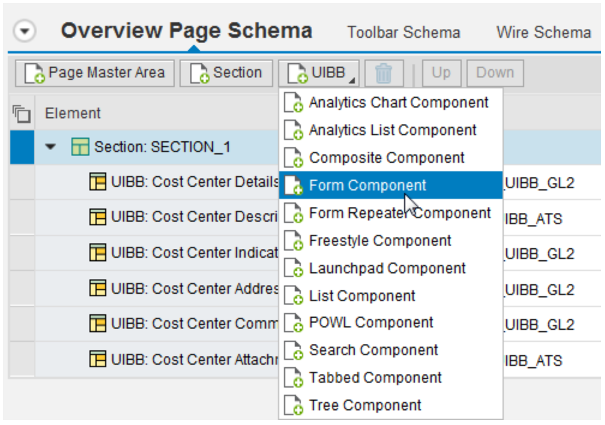
Advanced Settings:

Create Customizing

Config ID: MDGF_0G_CCTR_OVP

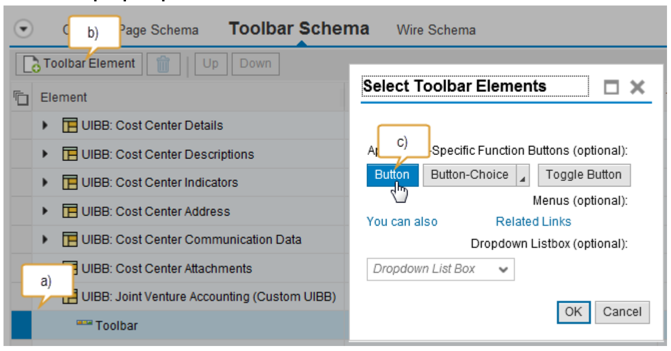
Description: Add CCTR JVA

9. Add the previously created form UIBB to the OVP.
   a. Select the OVP's Main Page MDGF_0G_CCTR_OVP in the Navigation pane.
   b. Collapse the Preview UIBB and expand the Overview Page Schema UIBB if required.
   c. Choose the Add UIBB button and select its context menu item Add Form Component.



   d. Define the attributes of your custom UIBB.
      i. Component = FPM_FORM_UIBB_GL2
      ii. Rendering Type = With Panel
      iii. Window Name = FORM_WINDOW
      iv. Config ID = ZMDGF_0G_CCTR_JVA
      v. Title = <Any Custom Title>
      vi. Column = 1 (relevant only if the OVP layout uses multiple columns)
      vii. Sequence Index = 7 (or a different position of your choice)
      viii. Container Stretching = UIBB does not need surrounding Containers to be stretched
   e. Save your changes.

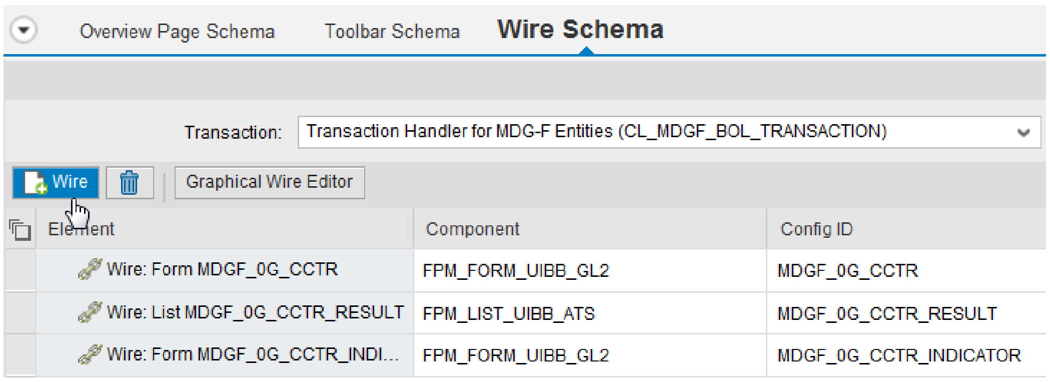
10. The UIBB is now part of the OVP. Still it needs an edit button the enable changing the field values.
11. Switch to tab Toolbar Schema.
12. Define an Edit button for your custom UIBB.
    a. Locate the UIBB in the toolbar schema list and select its Toolbar.
    b. Choose the Add Toolbar Element button.
    c. On the pop-up Select Toolbar Elements choose button "Button".


b)
Page Schema
Toolbar Schema
Wire Schema
Toolbar Element
Up
Down
Element
UIBB: Cost Center Details
UIBB: Cost Center Descriptions
UIBB: Cost Center Indicators
UIBB: Cost Center Address
UIBB: Cost Center Communication Data
UIBB: Cost Center Attachments
a)
UIBB: Joint Venture Accounting (Custom UIBB)
Toolbar
Select Toolbar Elements
c)
Specific Function Buttons (optional):
Button
Button-Choice
Toggle Button
Menus (optional):
You can also
Related Links
Dropdown Listbox (optional):
Dropdown List Box
OK
Cancel

13. Define the button's attributes.
    a. Text = Edit

b. Image Source = ~Icon/Edit
c. FPM Event ID = FPM_LOCAL_EDIT (use the F4 Help for selection)
d. Save your changes.

14. The edit button is available.
15. Finally, it is required to add the UIBB into the OVP's wire schema. Otherwise the UIBB would remain read-only all the time.
16. Switch to tab Wire Schema.
17. Choose the Add Wire button.



Transaction: Transaction Handler for MDG-F Entities (CL_MDGF_BOL_TRANSACTION)

Element	Component	Config ID
Wire: Form MDGF_0G_CCTR	FPM_FORM_UIBB_GL2	MDGF_0G_CCTR
Wire: List MDGF_0G_CCTR_RESULT	FPM_LIST_UIBB_ATS	MDGF_0G_CCTR_RESULT
Wire: Form MDGF_0G_CCTR_INDI...	FPM_FORM_UIBB_GL2	MDGF_0G_CCTR_INDICATOR

18. Define the wire's attributes.
    a. Use the F4 Help on attribute Config ID and select your UIBB.
    b. Use the F4 Help on attribute Source Config Name and select UIBB MDGF_0G_CCTR.
    c. Maintain attribute Connector Class as CL_FPM_CONNECTOR_BOL_IDENTITY.
    d. Use the F4 Help on attribute Port Identifier and select LS – Lead Selection.
    e. Save your changes.

The custom UIBB for Joint Venture Accounting is added to the OVP. If you start the cost center user interface the UIBB is visible. You might notice that some fields do not yet offer a value help. This implementation is realized by the next step.

4.3.7. Implement OVS search helps in the feeder class

The OVS search help is suitable for the fields Joint Venture, Recovery Indicator, Equity Type, and JIB/JIBE classes since all relate on additional values for displaying the search help correctly. Implementing the OVS consist of multiple methods that must be redefined accordingly.

1. Redefine method IF_FPM_GUIBB_FORM~GET_DEFINITION to add the feeder class as OVS search help implementation for the fields.
    a. Inherit from the parent implementation.
    b. Change the field attributes for the above-mentioned fields.
        i. Add the current class as OVS implementation.
        ii. Allow the deletion (clearance) of the field values.
    c. The source code is available in the appendix.
2. Create method OVS_OUTPUT_ZCTRVNAME to implement the search for the Joint Venture.
    a. Use a protected instance method with the description OVS Search: Joint Venture.
    b. Define the following parameters:
        i. Importing IR_QUERY_PARAMETER type ref to DATA as "User Input to be searched for".
        ii. Importing IV_FIELD_NAME type NAME_KOMP as "Field triggering the OVS".
        iii. Exporting ER_OUTPUT type ref to DATA as "Search Result".
    c. Prepare the result reference.
        i. You can use the generic type usmdz10_ts_ovs_output.
        ii. This type displays a simple table of name-value pairs.
    d. Check that the class attribute for the company code contains a value.
        i. The company code is mandatory for the selection of values.
        ii. If no company code is defined yet, show an appropriate error message and skip the search.

  e. Select the values from the customizing table(s).
  f. Handle the result.
     i. Map the selected values into the result reference.
     ii. Filter the selected values according to the user's search input.
     iii. …
  g. The source code is available in the appendix.

3. Repeat step 2 for the fields listed below accordingly.
   a. Method OVS_OUTPUT_ZCTRJIBCL for the JIB/JIBE Class.
   b. Method OVS_OUTPUT_ZCTRRECID for the Recovery Indicator.
4. Field Equity Type requires a more complex OVS since the field relates to both the company code and the actual joint venture. The example shows the power of OVS. It implements the following use cases:
   a. If the Joint Venture is not yet defined, the search displays all valid combinations of joint ventures and equity types. Selecting a single result sets both values in the user interface.
   b. If the Joint Venture is already defined, the search displays only those equity types that are defined for the current joint venture.
   c. Define each a protected structure and table type for the OVS as class types.
      i. TY_S_OVS_ZCTRETYPE as "OVS Structure for Equity Types"
```
TYPES:
  BEGIN OF ty_s_ovs_zctretype,
    vname     TYPE jv_name,
    vname_txt TYPE usmdz10_description,
    etype     TYPE jv_etype,
    etype_txt TYPE usmdz10_description,
  END OF ty_s_ovs_zctretype .
```
      ii. TY_TS_OVS_ZCTRETYPE as "OVS sorted table type for Equity Types"
```
TYPES:
  ty_ts_ovs_zctretype TYPE SORTED TABLE OF ty_s_ovs_zctretype
    WITH UNIQUE KEY vname etype .
```
   d. Define a protected instance class attribute MV_JOINT_VENTURE of type JV_NAME as "Current Joint Venture".
   e. Redefine method IF_FPM_GUIBB_OVS~HANDLE_PHASE_1 to check the current value of the joint venture.
      i. If a value is set, buffer it in MV_JOINT_VENTURE. Call the parent for further processing.
      ii. If no value is set, clear buffer MV_JOINT_VENTURE. Define the input structure for the OVS search using structure type TY_S_OVS_ZCTRETYPE.
      iii. The source code is available in the appendix.
   f. Create method OVS_OUTPUT_ZCTRETYPE to implement the search for the Equity type.
      i. Use a protected instance method with the description OVS Search: Equity Types.
      ii. Define the following parameters:
         1. Importing IR_QUERY_PARAMETER type ref to DATA as "User Input to be searched for".
         2. Importing IV_FIELD_NAME type NAME_KOMP as "Field triggering the OVS".
         3. Exporting ER_OUTPUT type ref to DATA as "Search Result".
      iii. Prepare the result reference according to the current joint venture.
         1. If a joint venture is defined, use the generic type usmdz10_ts_ovs_output.
         2. If no joint venture is defined, use the previously created type TY_TS_OVS_ZCTRETYPE.
      iv. Check that the class attribute for the company code contains a value.
         1. The company code is mandatory for the selection of values.
         2. If no company code is defined yet, show an appropriate error message and skip the search.
      v. Select the values from the customizing table.
         1. If a joint venture is defined, select all equity types belonging to the joint venture.

            2. If no joint venture is defined, select all combinations of joint ventures and equity types.
        vi. Map the selected values into the result reference.
        vii. The source code is available in the appendix.
    g. Redefine method OVS_HANDLE_PHASE_3 to transfer the user's selection of the complex OVS search help to the related fields in the user interface.
        i. If a joint venture is defined, the user has selected the equity type only.
        ii. If a joint venture is not defined, the user has selected both the joint venture and an equity type.
        iii. The source code is available in the appendix.
5. Implement the OVS for field JIB/JIBE Subclass A in a similar way as done for field Equity Type .
    a. Define each a protected structure and table type for the OVS as class types.
        i. TY_S_OVS_ZCTRJIBSA as "OVS Structure for Subclasses A"
```
TYPES:
  BEGIN OF ty_s_ovs_zctrjibsa,
    jibcl     TYPE jv_jibcl,
    jibcl_txt TYPE usmdz10_description,
    jibsa     TYPE jv_jibsa,
    jibsa_txt TYPE usmdz10_description,
  END OF ty_s_ovs_zctrjibsa .
```
        ii. TY_TS_OVS_ZCTRJIBSA as "OVS sorted table type for Subclasses A"
```
TYPES:
  ty_ts_ovs_zctrjibsa TYPE SORTED TABLE OF ty_s_ovs_zctrjibsa
    WITH UNIQUE KEY jibcl jibsa .
```
    b. Define a protected instance class attribute MV_CLASS of type JV_JIBCL as "Current Class".
    c. The source code is available in the appendix.
6. Redefine method OVS_HANDLE_PHASE_2 to call the previously created methods if an OVS search is triggered for the related fields.
    a. Implement a case statement on the field name.
    b. Dispatch the Joint Venture Accounting fields to their specific methods.
    c. Call the parent class for all other fields.
    d. The source code is available in the appendix.
    e. Save and activate your implementation.

The OVS is implemented and working for the Joint Venture Accounting fields.

4.3.8. Optional: Implement the text retrieval for transient text fields in the feeder class

If you have added the transient description fields to your user interface, some coding is required to provide the related description.

1. Redefine method GET_ENTITY_DATA to fill the descriptions into the transient fields.
    a. Since all fields require the company code, check if it is already defined. If not, skip processing.
    b. Determine the descriptions according to the given key values.

5. REMARKS ON DATA TRANSFER

Data transfer in MDG is separated into the following phases:

1. The Initial Load of master data from client system to the MDG system.
2. The Data Replication of master data from the MDG system to its client system

MDG-F provides standard content for both. Unfortunately, this content cannot cover custom fields. If your enhancements are part of data transfer, additional steps are required to ensure a fully working scenario. These steps strongly depend on the actual data transfer technology that you are using for your system landscape.

5.1.1. Initial Load using the MDM Generic Extractor

MDMGX already extracts Joint Venture fields. The standard MDMGX content for cost centers creates an XML file that contains the current database state.

Note that this is not always the case for other entity types in MDG-F. Some extract only a limited set of fields. In that case, you must enhance the selection criteria of MDMGX (refer to the related how-to guide for further details).

The import of the extracted XML file into the MDG system using the Data Import Framework (DIF) does not cover any custom enhancements. The following steps are required to enhance the functionality:

1. Logon to your MDG system.
2. Ensure that you have already enhanced the data model 0G and that you have re-generated all affected structures.
3. Start transaction SE80 and locate package USMD_MASS_LOAD.
4. Open the menu item Transformations.
    a. The import process consists of a single step that maps the content of the extracted XML file into an ABAP DDIC structure. The structures relate to the MDG generated structures. The transformation defines the mapping rules, basically which XML element is mapped to which structure component.
5. Open the transformation MDG_COST_CENTER.
    a. The code of a transformation is built according to the actual XML structure.
    b. All XML elements of type <COST_CENTERS> are processed within a loop <tt:loop name="LINE" ref=".RESULT">.
    c. A single XML file that contains data is mapped by a specific statement. The following code maps the value of the XML element <KOKRS> to the structure component COAREA – this is the mapping for the controlling area, of course.
    ```
    <KOKRS>
      <tt:value ref="$LINE.COAREA"/>
    </KOKRS>
    ```
    d. Fields that are currently not supported by the MDG data model 0G are simply ignored due to the skip statement, for example, the joint venture name is ignored with <tt:skip name="VNAME"/>.
    e. An extension or modification of the pre-defined transformations is not recommended. Proceed with the next step to create your custom transformation.
6. Right-click the pre-defined transformation name in the tree menu and choose Copy… in the context menu.
7. Choose a new name in the customer namespace, e.g. ZMDG_COST_CENTER.
8. Enhance the custom transformation according to your needs.
    a. It is important to keep the actual sequence of the XML file in the transformation.
    b. Do not change the sequence of XML elements. This will cause a failure during file processing.
    c. Omit fields that you do not need using the skip statement.
    d. Save and activate your custom transformation.
9. The final steps add your custom transformation to the data import framework.
10. Run transaction MDMIMG.
11. Navigate to Master Data Governance ➔ General Settings ➔ Data Transfer and start activity Define Object Types for Data Transfer.

12. Mark the object type that you have enhanced (e.g. CCTR Cost Center) and double-click on node Detailed Information for Object Types.
13. Replace the entry in column Transformations with your custom transformation (e.g. MDG_COST_CENTER with ZMDG_COST_CENTER).
14. Save your changes.

5.1.2. Data Replication

Joint Venture fields are already part of the cost center IDoc type COSMAS. The standard implementation for the MDG-F IDoc outbound for cost centers already supports the replication.

Joint Venture fields are not part of the cost center SOA service CostCentreReplicationBulkRequest. If the SOA Service is used for data replication, you are required to extend the SOA Service is for both the MDG-F outbound and for the SAP ECC inbound.

Other custom fields always require the extension of the desired data replication technology.

6. APPENDIX – SOURCE CODE OF ZCL_MDGF_GUIBB_CCTR_JVA

6.1.1. Method CREATE_STRUCT_RTTI

```
METHOD create_struct_rtti.

*! This method is used to enhance the field catalog.

*  It adds transient text fields '<FIELD_NAME>__TXT' for the following

*  attributes:

*  ZCTRETYPE  Equity Type

*  ZCTRJIBCL  JIB/JIBE Class

*  ZCTRJIBSA  JIB/JIBE Subclass A

*  ZCTRRECID  Recovery Indicator

*  ZCTRVNAME  Joint Venture

  DATA:

    lt_components TYPE cl_abap_structdescr=>component_table.


  FIELD-SYMBOLS:

    <ls_component> LIKE LINE OF lt_components.


  "Inherit the field catlog of the cost center. The catalog already

  "contains the input fields for Joint Venture Accounting.

  super->create_struct_rtti( ).

  lt_components = me->mo_struct_rtti->get_components( ).


  "Add the transient text fields one after the other as defined above.

  "Text fields use a string as data element.

  APPEND INITIAL LINE TO lt_components ASSIGNING <ls_component>.

  <ls_component>-name = |ZCTRETYPE{ cl_usmd_generic_genil_text=>gv_text_suffix }|.

  <ls_component>-type = cl_abap_elemdescr=>get_string( ).

  APPEND INITIAL LINE TO lt_components ASSIGNING <ls_component>.

  <ls_component>-name = |ZCTRJIBCL{ cl_usmd_generic_genil_text=>gv_text_suffix }|.

  <ls_component>-type = cl_abap_elemdescr=>get_string( ).

  APPEND INITIAL LINE TO lt_components ASSIGNING <ls_component>.
  <ls_component>-name = |ZCTRJIBSA{ cl_usmd_generic_genil_text=>gv_text_suffix }|.

  <ls_component>-type = cl_abap_elemdescr=>get_string( ).

  APPEND INITIAL LINE TO lt_components ASSIGNING <ls_component>.

  <ls_component>-name = |ZCTRRECID{ cl_usmd_generic_genil_text=>gv_text_suffix }|.

  <ls_component>-type = cl_abap_elemdescr=>get_string( ).

  APPEND INITIAL LINE TO lt_components ASSIGNING <ls_component>.

  <ls_component>-name = |ZCTRVNAME{ cl_usmd_generic_genil_text=>gv_text_suffix }|.

  <ls_component>-type = cl_abap_elemdescr=>get_string( ).


  "Update the field catalog.

  me->mo_struct_rtti = cl_abap_structdescr=>create( lt_components ).

ENDMETHOD.
```

6.1.2. Method GET_DEFINITION

```
METHOD if_fpm_guibb_form~get_definition.

*! This method is used to change the field definition for the Joint Venture

*  Accounting fields that shall use a OVS search help.

*  ZCTRETYPE  Equity Type

*  ZCTRJIBCL  JIB/JIBE Class

*  ZCTRJIBSA  JIB/JIBE Subclass A

*  ZCTRRECID  Recovery Indicator

*  ZCTRVNAME  Joint Venture

  FIELD-SYMBOLS:

    <ls_field_description> LIKE LINE OF et_field_description.


  "Inherit from parent.

  super->if_fpm_guibb_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 ).

  "Change the fields to add the current feeder class as OVS implementation
  "and allow the deletion of the field values.
  LOOP AT et_field_description ASSIGNING <ls_field_description>.
    CASE <ls_field_description>-name.
      WHEN 'ZCTRETYPE' "Equity Type
        OR 'ZCTRJIBCL' "JIB/JIBE Class
        OR 'ZCTRJIBSA' "JIB/JIBE Subclass A
        OR 'ZCTRRECID' "Recovery Indicator
        OR 'ZCTRVNAME'. "Joint Venture
        <ls_field_description>-ovs_name = me->mv_my_classname.
        <ls_field_description>-is_nullable = abap_true.
      WHEN OTHERS.
        CONTINUE.
    ENDCASE.
  ENDLOOP.
ENDMETHOD.
```

6.1.3. Method GET_ENTITY_DATA

```
METHOD get_entity_data.
*! This method is used to read the entity. It adds the texts for fields
*  having search or value helps.
  DATA:
    lv_class      TYPE jv_jibcl,
    lv_text_field TYPE name_komp.

  FIELD-SYMBOLS:
```

```
  <lv_key>  TYPE any,
  <lv_text> TYPE any.

"Inherit from the parent first. This set all key values.
super->get_entity_data(
  EXPORTING
    io_access = io_access
  CHANGING
    cs_data   = cs_data ).

"Check for the company code.
IF me->mv_company_code IS INITIAL.
  RETURN.
ENDIF.

"Set the joint venture description
lv_text_field = 'ZCTRVNAME' && cl_usmd_generic_genil_text=>gv_text_suffix.
ASSIGN COMPONENT:
  'ZCTRVNAME' OF STRUCTURE cs_data TO <lv_key>,
  lv_text_field OF STRUCTURE cs_data TO <lv_text>.
IF <lv_key> IS ASSIGNED AND <lv_key> IS NOT INITIAL
  AND <lv_text> IS ASSIGNED.
  SELECT SINGLE vtext FROM t8jvt INTO <lv_text>
    WHERE spras = sy-langu
      AND bukrs = me->mv_company_code
      AND vname = <lv_key>.
  UNASSIGN: <lv_key>, <lv_text>.
ENDIF.

"Set the equity type description
lv_text_field = 'ZCTRETYPE' && cl_usmd_generic_genil_text=>gv_text_suffix.
```

```
ASSIGN COMPONENT:
  'ZCTRETYPE' OF STRUCTURE cs_data TO <lv_key>,
  lv_text_field OF STRUCTURE cs_data TO <lv_text>.
IF <lv_key> IS ASSIGNED AND <lv_key> IS NOT INITIAL
  AND <lv_text> IS ASSIGNED.
  SELECT SINGLE etext FROM t8jet INTO <lv_text>
    WHERE spras = sy-langu
      AND bukrs = me->mv_company_code
      AND etype = <lv_key>.
  UNASSIGN: <lv_key>, <lv_text>.
ENDIF.

"Set the recovery indicator description
lv_text_field = 'ZCTRRECID' && cl_usmd_generic_genil_text=>gv_text_suffix.
ASSIGN COMPONENT:
  'ZCTRRECID' OF STRUCTURE cs_data TO <lv_key>,
  lv_text_field OF STRUCTURE cs_data TO <lv_text>.
IF <lv_key> IS ASSIGNED AND <lv_key> IS NOT INITIAL
  AND <lv_text> IS ASSIGNED.
  SELECT SINGLE ttext FROM t8jjt INTO <lv_text>
    WHERE spras = sy-langu
      AND bukrs = me->mv_company_code
      AND recid = <lv_key>.
  UNASSIGN: <lv_key>, <lv_text>.
ENDIF.

"Set the class description
lv_text_field = 'ZCTRJIBCL' && cl_usmd_generic_genil_text=>gv_text_suffix.
ASSIGN COMPONENT:
  'ZCTRJIBCL' OF STRUCTURE cs_data TO <lv_key>,
  lv_text_field OF STRUCTURE cs_data TO <lv_text>.
  IF <lv_key> IS ASSIGNED AND <lv_key> IS NOT INITIAL
    AND <lv_text> IS ASSIGNED.
    SELECT SINGLE ctext FROM t8j6b INTO <lv_text>
      WHERE spras = sy-langu
        AND bukrs = me->mv_company_code
        AND class = <lv_key>.
    lv_class = <lv_key>.
    UNASSIGN: <lv_key>, <lv_text>.
  ENDIF.


  "Set the subclass description
  IF lv_class IS NOT INITIAL.
    lv_text_field = 'ZCTRJIBSA' && cl_usmd_generic_genil_text=>gv_text_suffix.
    ASSIGN COMPONENT:
      'ZCTRJIBSA' OF STRUCTURE cs_data TO <lv_key>,
      lv_text_field OF STRUCTURE cs_data TO <lv_text>.
    IF <lv_key> IS ASSIGNED AND <lv_key> IS NOT INITIAL
      AND <lv_text> IS ASSIGNED.
      SELECT SINGLE ctext FROM t8j6d INTO <lv_text>
        WHERE spras    = sy-langu
          AND bukrs    = me->mv_company_code
          AND class    = lv_class
          AND subclass = <lv_key>.
      UNASSIGN: <lv_key>, <lv_text>.
    ENDIF.
  ENDIF.
ENDMETHOD.
```

6.1.4. Method HANDLE_PHASE_1

```
METHOD if_fpm_guibb_ovs~handle_phase_1.
*! This method is reflects the first OVS phase. It is used to prepare
*  the OVS search.
*  Using the parameter IO_OVS_CALLBACK it is possible to determine
*  values entered by the user that might be helpful for the search. In
*  addition it is possible to define a specifc search input structure
*  for complex search helps.
  DATA:
    ls_search_equity   TYPE ty_s_ovs_zctretype,
    ls_search_subclass TYPE ty_s_ovs_zctrjibsa.

  "A redefinition of the common logic is only needed for a few fields.
  CASE iv_field_name.
    WHEN 'ZCTRETYPE'. "Equity Type
      "The equity type depends on the joint venture. Its value is
      "determined using the callback parameter.
      CLEAR me->mv_joint_venture.
      io_ovs_callback->context_element->get_attribute(
        EXPORTING
          name  = 'ZCTRVNAME'
        IMPORTING
          value = me->mv_joint_venture ).
      IF me->mv_joint_venture IS INITIAL.
        "The joint venture is not yet selected. The OVS shall allow
        "searching for all valid combinations of joint ventures and
        "equity types.
        "Check for a pre-defined equity type
        io_ovs_callback->context_element->get_attribute(
          EXPORTING
            name  = 'ZCTRVNAME'
          IMPORTING
            value = ls_search_equity-etype ).
        "Set the input structure.
        io_ovs_callback->set_input_structure(
          EXPORTING
            input                    = ls_search_equity
            display_values_immediately = abap_true ).
      ELSE.
        "The joint venture is already selected. It is sufficient to use
        "the common OVS as implemented by the general MDG-F form feeder.
        super->if_fpm_guibb_ovs~handle_phase_1(
          iv_field_name   = iv_field_name
          io_ovs_callback = io_ovs_callback ).
      ENDIF.

    WHEN 'ZCTRJIBSA'. "Subclass A
      "The subclass A depends on the class. Its value is determined
      "using the callback parameter.
      CLEAR me->mv_class.
      io_ovs_callback->context_element->get_attribute(
        EXPORTING
          name  = 'ZCTRJIBCL'
        IMPORTING
          value = me->mv_class ).
      IF me->mv_class IS INITIAL.
        "The class is not yet selected. The OVS shall allow searching
        " for all valid combinations of classes and subclasses.
        "Check for a pre-defined subclass
        io_ovs_callback->context_element->get_attribute(
          EXPORTING
            name  = 'ZCTRJIBSA'
          IMPORTING
            value = ls_search_subclass-jibsa ).
        "Set the input structure.
        io_ovs_callback->set_input_structure(
            EXPORTING
              input                    = ls_search_subclass
              display_values_immediately = abap_true ).
          ELSE.
            "The class is already selected. It is sufficient to use the
            "common OVS as implemented by the general MDG-F form feeder.
            super->if_fpm_guibb_ovs~handle_phase_1(
              iv_field_name   = iv_field_name
              io_ovs_callback = io_ovs_callback ).
          ENDIF.

        WHEN OTHERS.
          "The parent implementation is re-usable for all other fields.
          super->if_fpm_guibb_ovs~handle_phase_1(
            iv_field_name   = iv_field_name
            io_ovs_callback = io_ovs_callback ).
      ENDCASE.
    ENDMETHOD.
```

6.1.5. Method OVS_HANDLE_PHASE_2

```
METHOD ovs_handle_phase_2.
*! This method is triggered by an OVS search in the user interface. It
*  dispatches the joint venture accounting fields to their specific
*  methods.
  CASE iv_field_name.
    WHEN 'ZCTRETYPE'. "Equity Type
      me->ovs_output_zctretype(
        EXPORTING
          ir_query_parameter = ir_query_parameter
          iv_field_name      = iv_field_name
        IMPORTING
          er_output          = er_output ).
```

```
WHEN 'ZCTRJIBCL'. "JIB/JIBE Class
  me->ovs_output_zctrjibcl(
    EXPORTING
      ir_query_parameter = ir_query_parameter
      iv_field_name      = iv_field_name
    IMPORTING
      er_output          = er_output ).

WHEN 'ZCTRJIBSA'. "JIB/JIBE Subclass A
  me->ovs_output_zctrjibsa(
    EXPORTING
      ir_query_parameter = ir_query_parameter
      iv_field_name      = iv_field_name
    IMPORTING
      er_output          = er_output ).

WHEN 'ZCTRRECID'. "Recovery Indicator
  me->ovs_output_zctrrecid(
    EXPORTING
      ir_query_parameter = ir_query_parameter
      iv_field_name      = iv_field_name
    IMPORTING
      er_output          = er_output ).

WHEN 'ZCTRVNAME'. "Joint Venture
  me->ovs_output_zctrvname(
    EXPORTING
      ir_query_parameter = ir_query_parameter
      iv_field_name      = iv_field_name
    IMPORTING
```

```
        er_output          = er_output ).


    WHEN OTHERS.
      "Invoke the parent.
      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.
```

6.1.6. Method OVS_HANDLE_PHASE_3

```
METHOD ovs_handle_phase_3.
*! This method is the phase 3 handler for OVS search helps.
*  It is used to set the user selection back to the related UI fields.
*  A general implementation is available in the parent class. This
*  method handles some of the Joint Venture Accounting fields only.
  DATA:
    lv_component_1 TYPE name_komp,
    lv_component_2 TYPE name_komp,
    lv_fieldname   TYPE name_komp.


  FIELD-SYMBOLS:
    <ls_field_value> LIKE LINE OF et_field_value,
    <ls_selection>   TYPE any,
    <lv_value>       TYPE any.
```

```
"get selection
IF ir_selection IS NOT BOUND.
  RETURN.
ENDIF.
ASSIGN ir_selection->* TO <ls_selection>.


IF iv_field_name EQ 'ZCTRETYPE'
  AND me->mv_joint_venture IS INITIAL.
  "Prepare special handling for equity types and joint venture.
  lv_component_1 = 'VNAME'.
  lv_component_2 = 'ETYPE'.
  lv_fieldname = 'ZCTRVNAME'.
ELSEIF iv_field_name EQ 'ZCTRJIBSA'
  AND me->mv_class IS INITIAL.
  "Prepare special handling for class and subclass.
  lv_component_1 = 'JIBCL'.
  lv_component_2 = 'JIBSA'.
  lv_fieldname = 'ZCTRJIBCL'.
ELSE.
  "All others do not require a specific handling.
  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 ).
  RETURN.
ENDIF.


"Transfer the dependent field (joint ventur or class)
```

```
"Map key to target field
APPEND INITIAL LINE TO et_field_value ASSIGNING <ls_field_value>.
CHECK sy-subrc EQ 0.
<ls_field_value>-name = lv_fieldname.
ASSIGN COMPONENT lv_component_1 OF STRUCTURE <ls_selection> TO <lv_value>.
CHECK sy-subrc EQ 0.
GET REFERENCE OF <lv_value> INTO <ls_field_value>-value.
UNASSIGN: <ls_field_value>, <lv_value>.

"map text to target field
lv_component_1 = lv_component_1 && '_TXT'.
ASSIGN COMPONENT lv_component_1 OF STRUCTURE <ls_selection> TO <lv_value>.
CHECK sy-subrc EQ 0.
APPEND INITIAL LINE TO et_field_value ASSIGNING <ls_field_value>.
CHECK sy-subrc EQ 0.
<ls_field_value>-name = lv_fieldname && cl_usmd_generic_genil_text=>gv_text_suffix.
GET REFERENCE OF <lv_value> INTO <ls_field_value>-value.
UNASSIGN: <ls_field_value>, <lv_value>.

"Transfer the OVS field (equity type or subclass)
"Map key to target field
APPEND INITIAL LINE TO et_field_value ASSIGNING <ls_field_value>.
CHECK sy-subrc EQ 0.
<ls_field_value>-name = iv_field_name.
ASSIGN COMPONENT lv_component_2 OF STRUCTURE <ls_selection> TO <lv_value>.
CHECK sy-subrc EQ 0.
GET REFERENCE OF <lv_value> INTO <ls_field_value>-value.
UNASSIGN: <ls_field_value>, <lv_value>.

"map text to target field
lv_component_2 = lv_component_2 && '_TXT'.
```

```
  UNASSIGN: <ls_field_value>, <lv_value>.

  ASSIGN COMPONENT lv_component_2 OF STRUCTURE <ls_selection> TO <lv_value>.

  CHECK sy-subrc EQ 0.

  APPEND INITIAL LINE TO et_field_value ASSIGNING <ls_field_value>.

  CHECK sy-subrc EQ 0.

  <ls_field_value>-name = iv_field_name && cl_usmd_generic_genil_text=>gv_text_suffix.

  GET REFERENCE OF <lv_value> INTO <ls_field_value>-value.

  UNASSIGN: <ls_field_value>, <lv_value>.

ENDMETHOD.
```

6.1.7. Method OVS_OUTPUT_ZCTRETYPE

```
METHOD ovs_output_zctretype.

*! This method implements the OVS search for the Equity Type. The data

*  retrieval depends on the current joint venture. If no value is

*  defined yet, the search has to return all combinations of joint

*  ventures and equity types. If a value is selected, only those equity

*  type may be returned that belong to the joint venture.

  DATA:

    lo_message_container TYPE REF TO cl_crm_genil_global_mess_cont,

    ls_combined          TYPE ty_s_ovs_zctretype,

    ls_equity_type       TYPE usmdz10_s_ovs_output,

    ls_joint_venture     TYPE usmdz10_s_ovs_output,

    ls_t8jet             TYPE t8jet,

    ls_t8jvt             TYPE t8jvt,

    lt_equity_types      TYPE usmdz10_ts_ovs_output,

    lt_joint_ventures    TYPE usmdz10_ts_ovs_output.


  FIELD-SYMBOLS:

    <ls_output>   TYPE usmdz10_s_ovs_output,

    <lt_combined> TYPE ty_ts_ovs_zctretype,

    <lt_output>   TYPE usmdz10_ts_ovs_output.
```

```
"Prepare the output table. The reference must be returned even if
"the execution of the search does not return any records.
IF me->mv_joint_venture IS NOT INITIAL.
  "Common result
  CREATE DATA er_output TYPE usmdz10_ts_ovs_output.
  ASSIGN er_output->* TO <lt_output>.
ELSE.
  "All Equity Types of all Joint Ventures
  CREATE DATA er_output TYPE ty_ts_ovs_zctretype.
  ASSIGN er_output->* TO <lt_combined>.
ENDIF.

"Check that a company code has already been defined
IF me->mv_company_code IS INITIAL.
  "Raise an error and return.
  "Message: Enter a value for field company code and choose ENTER
  MESSAGE e010(usmdz10) INTO sy-ucomm.
  lo_message_container = cl_crm_bol_core=>get_instance( )->get_global_message_cont( ).
  ASSERT lo_message_container IS BOUND.
  lo_message_container->add_message(
    iv_msg_type       = sy-msgty
    iv_msg_id         = sy-msgid
    iv_msg_number     = sy-msgno
    iv_show_only_once = abap_true ).
  RETURN.
ENDIF.

"Select the values from the data base table.
IF me->mv_joint_venture IS NOT INITIAL.
  "Get Equity Types of the joint venture.
  SELECT etype etype FROM t8jg INTO TABLE <lt_output>
```

```
    WHERE bukrs = me->mv_company_code
      AND vname = me->mv_joint_venture.
  "TODO: FDATE vs. edition validity?

  "Try to add the Equity Type texts in the user's language.
  LOOP AT <lt_output> ASSIGNING <ls_output>.
    CLEAR ls_t8jet.
    SELECT SINGLE * FROM t8jet INTO ls_t8jet
      WHERE spras = sy-langu
        AND bukrs = me->mv_company_code
        AND etype = <ls_output>-key.
    IF ls_t8jet-etext IS NOT INITIAL.
      <ls_output>-text = ls_t8jet-etext.
    ENDIF.
  ENDLOOP.
ELSE.
  "Get all joint ventures first.
  SELECT vname vname FROM t8jv INTO TABLE lt_joint_ventures
    WHERE bukrs = mv_company_code.

  "Handle each joint venture individually.
  LOOP AT lt_joint_ventures INTO ls_joint_venture.
    "Try to add the Joint Venture texts in the user's language.
    CLEAR ls_t8jvt.
    SELECT SINGLE * FROM t8jvt INTO ls_t8jvt
      WHERE spras = sy-langu
        AND bukrs = me->mv_company_code
        AND vname = ls_joint_venture-key.
    IF ls_t8jvt-vtext IS NOT INITIAL.
      ls_joint_venture-text = ls_t8jvt-vtext.
    ENDIF.
```

```
      "Get the Equity Types for the current Joint Venture
      CLEAR lt_equity_types.
      SELECT etype etype FROM t8jg INTO TABLE lt_equity_types
        WHERE bukrs = me->mv_company_code
          AND vname = ls_joint_venture-key.
      "TODO: FDATE vs. edition validity?

      "Handle the result.
      LOOP AT lt_equity_types INTO ls_equity_type.
        "Try to add the Equity Type texts in the user's language.
        CLEAR ls_t8jet.
        SELECT SINGLE * FROM t8jet INTO ls_t8jet
          WHERE spras = sy-langu
            AND bukrs = me->mv_company_code
            AND etype = ls_equity_type-key.
        IF ls_t8jet-etext IS NOT INITIAL.
          ls_equity_type-text = ls_t8jet-etext.
        ENDIF.

        "Add the Equity Type to the result.
        CLEAR ls_combined.
        ls_combined-vname = ls_joint_venture-key.
        ls_combined-vname_txt = ls_joint_venture-text.
        ls_combined-etype = ls_equity_type-key.
        ls_combined-etype_txt = ls_equity_type-text.
        INSERT ls_combined INTO TABLE <lt_combined>.
      ENDLOOP.
    ENDLOOP.
  ENDIF.
ENDMETHOD.
```

6.1.8. Method OVS_OUTPUT_ZCTRJIBCL

```
METHOD ovs_output_zctrjibcl.

*! This method implements the OVS search for the JIB/JIBE Classes.

  DATA:

    lo_message_container TYPE REF TO cl_crm_genil_global_mess_cont,

    ls_description       TYPE t8j6b.


  FIELD-SYMBOLS:

    <ls_output> TYPE usmdz10_s_ovs_output,

    <lt_output> TYPE usmdz10_ts_ovs_output.


  "Prepare the output table. The reference must be returned even if

  "the execution of the search does not return any records.

  CREATE DATA er_output TYPE usmdz10_ts_ovs_output.

  ASSIGN er_output->* TO <lt_output>.


  "Check that a company code has already been defined

  IF me->mv_company_code IS INITIAL.

    "Raise an error and return.

    "Message: Enter a value for field company code and choose ENTER

    MESSAGE e010(usmdz10) INTO sy-ucomm.

    lo_message_container = cl_crm_bol_core=>get_instance( )->get_global_message_cont( ).

    ASSERT lo_message_container IS BOUND.

    lo_message_container->add_message(

      iv_msg_type       = sy-msgty

      iv_msg_id        = sy-msgid

      iv_msg_number     = sy-msgno

      iv_show_only_once = abap_true ).

    RETURN.

  ENDIF.
  "Select the values from the data base table
  SELECT class class FROM t8j6a INTO TABLE <lt_output>
    WHERE bukrs = mv_company_code.

  "Try to add the class texts in the user's language.
  LOOP AT <lt_output> ASSIGNING <ls_output>.
    CLEAR ls_description.
    SELECT SINGLE * FROM t8j6b INTO ls_description
      WHERE spras = sy-langu
        AND bukrs = me->mv_company_code
        AND class = <ls_output>-key.
    IF ls_description-ctext IS NOT INITIAL.
      <ls_output>-text = ls_description-ctext.
    ENDIF.
  ENDLOOP.

  "Filter results
  me->ovs_output_filter(
    EXPORTING
      iv_field_name      = iv_field_name
      ir_query_parameter = ir_query_parameter
    CHANGING
      cr_output          = er_output ).
ENDMETHOD.
```

6.1.9. Method OVS_OUTPUT_ZCTRJIBSA

```
METHOD ovs_output_zctrjibsa.
*! This method implements the OVS search for the JIB/JIBE Subclass A.
*  The data retrieval depends on the current joint JIB/JIBE Class. If
*  no value is defined yet, the search has to return all combinations
*  the classes and subclasses. If a value is selected, only those
*  subclasses may be returned that belong to the class.
  DATA:
    lo_message_container TYPE REF TO cl_crm_genil_global_mess_cont,
    ls_combined          TYPE ty_s_ovs_zctrjibsa,
    ls_class             TYPE usmdz10_s_ovs_output,
    ls_subclass          TYPE usmdz10_s_ovs_output,
    ls_t8j6d             TYPE t8j6d,
    ls_t8j6b             TYPE t8j6b,
    lt_classes           TYPE usmdz10_ts_ovs_output,
    lt_subclasses        TYPE usmdz10_ts_ovs_output.

  FIELD-SYMBOLS:
    <ls_output>   TYPE usmdz10_s_ovs_output,
    <lt_combined> TYPE ty_ts_ovs_zctrjibsa,
    <lt_output>   TYPE usmdz10_ts_ovs_output.

  "Prepare the output table. The reference must be returned even if
  "the execution of the search does not return any records.
  IF me->mv_class IS NOT INITIAL.
    "Common result
    CREATE DATA er_output TYPE usmdz10_ts_ovs_output.
    ASSIGN er_output->* TO <lt_output>.
  ELSE.
    "All Subclasses of all Classes
    CREATE DATA er_output TYPE ty_ts_ovs_zctrjibsa.
    ASSIGN er_output->* TO <lt_combined>.
  ENDIF.

  "Check that a company code has already been defined
  IF me->mv_company_code IS INITIAL.
    "Raise an error and return.
    "Message: Enter a value for field company code and choose ENTER
    MESSAGE e010(usmdz10) INTO sy-ucomm.
    lo_message_container = cl_crm_bol_core=>get_instance( )->get_global_message_cont( ).
    ASSERT lo_message_container IS BOUND.
    lo_message_container->add_message(
      iv_msg_type      = sy-msgty
      iv_msg_id        = sy-msgid
      iv_msg_number     = sy-msgno
      iv_show_only_once = abap_true ).
    RETURN.
  ENDIF.


  "Select the values from the data base table.
  IF me->mv_class IS NOT INITIAL.
    "Get Subclasses of the Class.
    SELECT subclass subclass FROM t8j6c INTO TABLE <lt_output>
      WHERE bukrs = me->mv_company_code
        AND class = me->mv_class.


    "Try to add the Subclass texts in the user's language.
    LOOP AT <lt_output> ASSIGNING <ls_output>.
      CLEAR ls_t8j6d.
      SELECT SINGLE * FROM t8j6d INTO ls_t8j6d
        WHERE spras = sy-langu
          AND bukrs = me->mv_company_code
          AND class    = me->mv_class
          AND subclass = <ls_output>-key.
      IF ls_t8j6d-ctext IS NOT INITIAL.
        <ls_output>-text = ls_t8j6d-ctext.
      ENDIF.
    ENDLOOP.
  ELSE.
    "Get all classes first.
    SELECT class class FROM t8j6a INTO TABLE lt_classes
      WHERE bukrs = mv_company_code.

    "Handle each joint venture individually.
    LOOP AT lt_classes INTO ls_class.
      "Try to add the Joint Venture texts in the user's language.
      CLEAR ls_t8j6b.
      SELECT SINGLE * FROM t8j6b INTO ls_t8j6b
        WHERE spras = sy-langu
          AND bukrs = me->mv_company_code
          AND class = ls_class-key.
      IF ls_t8j6b-ctext IS NOT INITIAL.
        ls_class-text = ls_t8j6b-ctext.
      ENDIF.

      "Get the Equity Types for the current Joint Venture
      CLEAR lt_subclasses.
      SELECT subclass subclass FROM t8j6c INTO TABLE lt_subclasses
        WHERE bukrs = me->mv_company_code
          AND class = ls_class-key.
      "TODO: FDATE vs. edition validity?

      "Handle the result.
      LOOP AT lt_subclasses INTO ls_subclass.
        "Try to add the Equity Type texts in the user's language.
        CLEAR ls_t8j6d.
        SELECT SINGLE * FROM t8j6d INTO ls_t8j6d
          WHERE spras = sy-langu
```

```
          AND bukrs = me->mv_company_code
          AND subclass = ls_subclass-key.
      IF ls_t8j6d-ctext IS NOT INITIAL.
        ls_subclass-text = ls_t8j6d-ctext.
      ENDIF.

      "Add the Equity Type to the result.
      CLEAR ls_combined.
      ls_combined-jibcl = ls_class-key.
      ls_combined-jibcl_txt = ls_class-text.
      ls_combined-jibsa = ls_subclass-key.
      ls_combined-jibsa_txt = ls_subclass-text.
      INSERT ls_combined INTO TABLE <lt_combined>.
    ENDLOOP.
  ENDLOOP.
 ENDIF.
ENDMETHOD.
```

6.1.10. Method OVS_OUTPUT_ZCTRRECID

```
METHOD ovs_output_zctrrecid.
*! This method implements the OVS search for the Recovery Indicator.
  DATA:
    lo_message_container TYPE REF TO cl_crm_genil_global_mess_cont,
    ls_description       TYPE t8jjt.

  FIELD-SYMBOLS:
    <ls_output> TYPE usmdz10_s_ovs_output,
    <lt_output> TYPE usmdz10_ts_ovs_output.

  "Prepare the output table. The reference must be returned even if
  "the execution of the search does not return any records.
  CREATE DATA er_output TYPE usmdz10_ts_ovs_output.
```

```
ASSIGN er_output->* TO <lt_output>.


"Check that a company code has already been defined
IF me->mv_company_code IS INITIAL.
  "Raise an error and return.
  "Message: Enter a value for field company code and choose ENTER
  MESSAGE e010(usmdz10) INTO sy-ucomm.
  lo_message_container = cl_crm_bol_core=>get_instance( )->get_global_message_cont( ).
  ASSERT lo_message_container IS BOUND.
  lo_message_container->add_message(
    iv_msg_type       = sy-msgty
    iv_msg_id         = sy-msgid
    iv_msg_number     = sy-msgno
    iv_show_only_once = abap_true ).
  RETURN.
ENDIF.


"Select the values from the data base table
SELECT recid recid FROM t8jj INTO TABLE <lt_output>
  WHERE bukrs = mv_company_code.


"Try to add the Joint Venture texts in the user's language.
LOOP AT <lt_output> ASSIGNING <ls_output>.
  CLEAR ls_description.
  SELECT SINGLE * FROM t8jjt INTO ls_description
    WHERE spras = sy-langu
      AND bukrs = me->mv_company_code
      AND recid = <ls_output>-key.
  IF ls_description-ttext IS NOT INITIAL.
    <ls_output>-text = ls_description-ttext.
  ENDIF.
  ENDLOOP.


  "Filter results
  me->ovs_output_filter(
    EXPORTING
      iv_field_name      = iv_field_name
      ir_query_parameter = ir_query_parameter
    CHANGING
      cr_output          = er_output ).
ENDMETHOD.
```

6.1.11. Method OVS_OUTPUT_ZCTRVNAME

```
METHOD ovs_output_zctrvname.
*! This method implements the OVS search for the Joint Venture.
  DATA:
    lo_message_container TYPE REF TO cl_crm_genil_global_mess_cont,
    ls_description       TYPE t8jvt.


  FIELD-SYMBOLS:
    <ls_output> TYPE usmdz10_s_ovs_output,
    <lt_output> TYPE usmdz10_ts_ovs_output.


  "Prepare the output table. The reference must be returned even if
  "the execution of the search does not return any records.
  CREATE DATA er_output TYPE usmdz10_ts_ovs_output.
  ASSIGN er_output->* TO <lt_output>.


  "Check that a company code has already been defined
  IF me->mv_company_code IS INITIAL.
    "Raise an error and return.
    "Message: Enter a value for field company code and choose ENTER
    MESSAGE e010(usmdz10) INTO sy-ucomm.
  lo_message_container = cl_crm_bol_core=>get_instance( )->get_global_message_cont( ).
  ASSERT lo_message_container IS BOUND.
  lo_message_container->add_message(
    iv_msg_type       = sy-msgty
    iv_msg_id         = sy-msgid
    iv_msg_number     = sy-msgno
    iv_show_only_once = abap_true ).
  RETURN.
ENDIF.

"Select the values from the data base table
SELECT vname vname FROM t8jv INTO TABLE <lt_output>
  WHERE bukrs = mv_company_code.

"Try to add the Joint Venture texts in the user's language.
LOOP AT <lt_output> ASSIGNING <ls_output>.
  CLEAR ls_description.
  SELECT SINGLE * FROM t8jvt INTO ls_description
    WHERE spras = sy-langu
      AND bukrs = me->mv_company_code
      AND vname = <ls_output>-key.
  IF ls_description-vtext IS NOT INITIAL.
    <ls_output>-text = ls_description-vtext.
  ENDIF.
ENDLOOP.

"Filter results
me->ovs_output_filter(
  EXPORTING
    iv_field_name      = iv_field_name
    ir_query_parameter = ir_query_parameter
    CHANGING
      cr_output          = er_output ).
ENDMETHOD.
```

7. ADDITIONAL INFORMATION

Further Reading

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

SAP Roadmap Explorer

- Please see the roadmap for SAP Master Data Governance

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 |

SAP Notes

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

Note Number	Note Description
3194967	MDG Customer Connection 2021 for S/4HANA 2022
3043582	MDG Customer Connection 2020
2221398	MDG-BP/C/S/CA: (Un-)Supported Fields in Data Model BP
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
3134600	MDG-M: Supported fields in Data Model MM
1806108	Functional restrictions in MDG-M in MDG7 (incl. SP02)
2129261	Functional restrictions in MDG-M in MDG8
2284745	Functional Restrictions in MDG for Material with SAP Master Data Governance 9.0
2461516	Functional Restrictions in MDG for Material with SAP Master Data Governance 9.1
2656693	Functional Restrictions in MDG for Material in SAP Master Data Governance 9.2 and on SAP S/4HANA 1809
2816571	Functional Restrictions in MDG for Material on SAP S/4HANA 1909
2948873	Functional Restrictions in MDG for Material on SAP S/4HANA 2020

2479869	Usage of Lean Classification with SAP Master Data Governance
3070012	Functional Restrictions in MDG for Material on SAP S/4HANA 2021
3219945	Functional Restrictions in MDG for Material on SAP S/4HANA 2022
1619534	How to Create, Enhance and Adapt FPM Applications

www.sap.com/contactsap

THE BEST RUN SAP®