PUBLIC

Extend the Business Partner - Overview

Applicable Releases:
All

Version 1.4

July 2020

Document History

Document Version	Description
1.0	First official release of this guide with MDG 6.0
1.1	Updates for MDG 6.1
1.2	Updates for MDG 7.0
1.3	Updates for MDG 9.1
1.4	Updates for MDG 1909

1. BUSINESS SCENARIO

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

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

This how to guide provides you with the foundation knowledge you need to know about business partner, customer and supplier data and their related governance solutions.

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
2105467	MDG Performance
2221398	MDG-BP/C/S/CA: (Un-)Supported Fields in Data Model BP
2561461	Scope of support for SAP Master Data Governance (MDG)
2847807	MDG-BP/C/S/CA: Usage of MDG Tools and Processes

Configuration and Enhancement of SAP Master Data Governance

Web Dynpro & Floorplan Manager

3. BACKGROUND INFORMATION

MDG offers different solutions in the business partner area:

- MDG-BP for Business Partners
- MDG-C for Customers
- MDG-S for Suppliers

MDG-BP is the foundation for MDG-C and MDG-S. Each customer and supplier are always based upon a business partner. You cannot create a customer or supplier without a corresponding business partner. MDG re-uses the common SAP Business Partner as available in each SAP system via transaction BP.

MDG-C is an extension of MDG-BP governance. It adds the specific tables of customer master data as available in each SAP S/4 HANA system in transaction BP using a customer role (or in old SAP ERP systems via transactions XD0*).

MDG-S is an extension of MDG-BP governance. It adds the specific tables of supplier master data as available in each SAP S/4 HANA system in transaction BP using a supplier (vendor) role (or in old SAP ERP systems via transactions XK0*).

From a technical perspective, the MDG solutions are implemented in two different software layers. The general business partner components are built in the MDG foundation (MDG_FND) layer whereas the customer and supplier components belong to the MDG application (MDG_APPL) layer. This is important to remember if you plan enhancements of your solutions.

4. FUNCTIONAL RESTRICTIONS

The following restrictions apply:

- All information published in the how to guide correspond to the current release of SAP MDG, Central Governance being available for public use.
- If your system is an older version of SAP MDG, Central Governance some functionality mentioned in this how to guide might not exist.

5. STEP BY STEP EXPLANATION

5.1. Data Model

The MDG data model for business partners, customers and suppliers is model BP. A detailed list of all fields being supported by the data model is provided by SAP note 2221398 MDG-BP/C/S/CA: (Un-)Supported Fields in Data Model BP.

Use report USMD_DISPLAY_DATA_MODEL with data model BP to display its current state in your local system.

The entity types of the model are mainly reuse entity types. The active data of the model is stored in existing database tables of the SAP system (for example the business partner is stored in table BUT000, the customer is stored in table KNA1, the vendor is stored in LFA1, and so on). The MDG staging tables are used for the inactive version of the data that is currently processed within a governance process.

5.1.1. Multiple Assignments

Multiple Assignments allow you to link one or more customers / suppliers to a single business partner. Within SAP MDG, the functionality exists since MDG 6.0. MDG client systems having the release version S/4 HANA 1809 or newer support the functionality, too.

5.1.2. Hierarchies

MDG-BP, MDG-C and MDG-S support the creation and maintenance of analytical hierarchies. The hierarchies are built using the business partner. MDG specific flexible entity types are used for modelling the hierarchies. These do not relate to any re-usable SAP Business Partner, Customer or Supplier hierarchy.

The data is solely stored in the generated MDG tables. Data replication is not available.

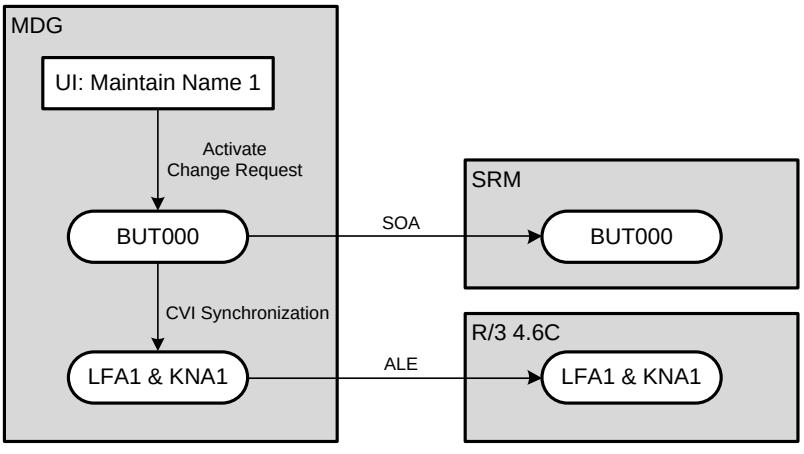
Data maintenance is supported either by the generic MDG user interfaces for collective and hierarchy processing, or via the enhancements of the single object maintenance user interfaces as introduced by MDG 9.0.

5.1.3. Customer / Vendor Integration (CVI)

Customer and supplier governance are both based on business partner governance. In MDG the business partner is the leading object. Using MDG, a business partner is always created.

MDG reuses the existing principles of the Customer / Vendor Integration (CVI). Depending on the CVI configuration and the entered data, a customer and/or supplier record is created in addition to the business partner.

Taking a detailed look at the related database tables it is obvious, that some fields are identical in the different entity types. CVI is a generic tool that can synchronize parts of the business partner master data with the related ERP customer and supplier master data and vice versa.



The figure shows the flow of the field Name 1 as example. The data is maintained in the business partner part of the user interface for either customer or supplier governance. As soon as the related change request is approved, data activation stores the field in the business partner table BUT000. This triggers CVI, which creates an additional entry in the related tables for customer master data (table KNA1) and vendor master data (table LFA1).

One benefit of this behavior is the simplified data replication. If the newly created MDG records are sent to an SRM system, the system uses the business partner data. SRM neither knows the customer nor the vendor since this is ERP specific data. If an (older) ERP system is the target receiver, it is possible to replicate either the customer data, or the vendor data, or a combination of both, in a direct way (for example, using the common DEBMAS or CREMAS IDocs). All records in all systems have the same value for Name 1.

CVI synchronizes the following segments and fields:

- General Data
    - Names
    - Search Terms
    - Trading Partner
    - Business Partner Category
        - ERP customer and supplier do not differentiate between organizations, persons and groups. Therefore, all business partners are treated as organizations.
- Address Data
    - The business partner can handle multiple addresses. This feature is not available for ERP's customer and supplier. Therefore, only the standard address is synchronized.
- Communication Data
    - Only address dependent communication data is synchronized.
- Bank Accounts
- Industry Sectors
    - The business partner can handle multiple industry systems and sectors. This feature is not available for ERP's customer and supplier. Therefore, only the standard industry sector of the standard industry system is synchronized.
- Tax Numbers
    - The business partner stores tax numbers in a table format. ERP customer and supplier use simple fields combined with a table for European VAT Numbers. CVI takes care of the correct synchronization of the business partner table with the fields and tables for customer or supplier.

When you consider an extension of MDG, you can choose from several options to add tables and fields. Your choice strongly depends upon the following question: Which entity receives the additional information?

The following scenarios are possible:

- The new field or table is available only in the business partner.
- The new field or table is available only in the customer or only in the supplier.
- The new field or table is available in both the business partner and in either the customer or in the supplier.

Each extension requires a specific sequence of steps that must be implemented correctly. Detailed how-to guides explain each extension option.

CVI is used by MDG. It is neither developed nor maintained by MDG. The synchronization happens during the activation of master data, and always in the following direction: from the business partner to the customer or the supplier. During the setup phase of MDG-C and/or MDG-S it is possible to use the CVI Synchronization Cockpit to create business partners from existing customers or vendors.

All MDG-C or MDG-S related use cases require a valid configuration of the CVI customizing. You can apply settings in Customizing via Cross-Application Components → Master Data Synchronization → Customer Vendor Integration.

MDG's core scenario of central master data governance requires that you set up Customizing to enable the synchronization direction BP to Customer / Supplier. This involves the following tasks:

- Defining at least one business partner role that triggers the creation of a customer / supplier.
    - It is possible to define multiple roles. If you do this, you must assign the correct role when maintaining the user interface. If you define only one role, the system uses the role automatically when it stores the business partner in the active area.
    - If you use either the Customer Like UI or the Vendor Like UI, you must define a unique role in customizing. If you add multiple roles, you need to either derive the default role to be used, or to add the list for business partner role maintenance in the UI's customizing.
- Defining a mapping of the business partner grouping to its related customer / supplier account group.
    - The MDG UIs do not enforce the maintenance of the BP grouping. If a user does not select any BP grouping, the system chooses a default grouping based on the business partner ID in use. If a business partner ID is maintained, the default grouping for external numbering is used. If no business partner ID is maintained, the default grouping for internal numbering is used. You must correctly configure the related BP groupings when applying customizing settings for CVI.
    - The group mapping defines the key mapping implicitly since each Business Partner / Customer / Supplier (account) group is assigned to a specific number range. Ensure that the respective number ranges match, especially if you use the same numbers for the BP and for the customer / supplier.
    - The first customer / supplier that is added to a BP is the "standard assignment". This assignment derives its account group according to configuration specified in customizing settings for CVI. The account group can only be changed manually in the MDG UIs if you have enabled the flexible grouping functionality. Additional assignments require the manual selection of the account group.
    - If you use either the Customer Like UI or the Vendor Like UI, only the following numbering combinations for BPs to customers / suppliers are allowed:
        - BP internal and C/S internal (results in different IDs)
        - BP internal and C/S external with same number (results in identical IDs)
        - BP external and C/S external with same number (results in identical IDs)
        - Other combinations would cause a deadlock in the UI.
- Making sure to check if further settings / mappings on segment level (e.g. industry sectors) are required.

MDG processes do not require that you set up the synchronization direction Customer / Supplier to BP. This direction is only useful if you want to transfer existing customers or suppliers to BPs. Techniques for doing this include using the MDS Load Cockpit functionality of CVI.

A detailed description of the CVI is available in the SAP Application Help.

5.2. Access Classes

Data model BP being a reuse area model requires the implementation of an access classes that is responsible for all actions related to the reuse area: read and store the data from and in the reuse area database tables, derivations, validations, etc.

MDG uses a single access class that distributes all calls to different, object-specific handler classes. This allows an extension of the access class simply by creating and registering a custom handler class. The access class is only responsible for controlling the calls. The actual execution of the calls, which involves, for example reading data or saving data, is implemented within the handler classes. Following the segregation of duty principle there is a handler for each object (in other words one for business partner, one for multiple assignment, one for customer, and one for supplier). This handler alone is fully responsible for the related object (in other words, the supplier handler must never change the customer object).

Access classes are required to route the calls by the MDG framework to the specific handler implementations. Since MDG-BP, MDG-C and MDG-S consist of multiple software layers, a single access class would not be enough for this need. Therefore, the implementation consists of three classes in a hierarchy with inheritance:

- CL_MDG_BS_BP_ACCESS_MASTER
    - The master class is the one being called by the MDG framework. Its object instance depends on the current software layer it is running in. In the MDG foundation layer, it is a reference of class CL_MDG_BS_FND_ACCESS whereas in the MDG application layer it is a reference of class CL_MDG_ECC_ACCESS. The class implements the MDG framework's access interface and provides some generic functionality.
- CL_MDG_BS_FND_ACCESS
    - This class is the MDG foundation layer implementation.
    - It inherits from class CL_MDG_BS_BP_ACCESS_MASTER .
    - It contains both generic implementations usable for all entity types of MDG-BP Customer Governance (MDG-C) and Supplier Governance (MDG-S) as well specific coding for all entity types of MDG-BP.
- CL_MDG_BS_ECC_ACCESS
    - This class is the MDG application layer implementation.
    - It inherits from class CL_MDG_BS_FND_ACCESS.
    - It contains only coding for all entity types of Customer Governance (MDG-C) and Supplier Governance (MDG-S).

If you want to create an extension of MDG, it is strictly not recommended to enhance existing access classes respectively to create an access class of your own. We recommend that you implement a specific handler class instead.

All calls from the access to the handler classes are triggered by the access class. Therefore, the access class collects a list of registered handlers consisting of both predefined SAP handlers as well as custom handlers first. Then the access class calls each handler within a loop. SAP owned handled classes are usually called before custom handler classes.

5.3. Handler Classes

The link between an access class and a handler class is the handler interface IF_MDG_BS_BP_ACCESS_HANDLER. A handler class must implement this interface. Since major parts of the application logic needed for MDG-BP, MDG-C, and MDG-S is the same for each object, two abstract classes provide this common implementation. In addition, object specific handler classes provide dedicated functionality only for the object the class is responsible for. all classes are implemented in a hierarchy with inheritance to support the needs of the different software layers:

- CL_MDG_BS_FND_HANDLER
  - CL_MDG_BS_BP_HANDLER
  - CL_MDG_BS_ECC_HANDLER
    - CL_MDG_BS_MLT_ASSGNMNT_HANDLER
    - CL_MDG_BS_CUST_HANDLER
    - CL_MDG_BS_SUPPL_HANDLER

All classes strictly follow the segregation of duty principle. A single handler is responsible for a single object only.

If you want to extend MDG by creating your own handler classes, there are several possibilities. A general recommendation for the creation of a custom handler class is that your new class inherits from either the abstract foundation handler or from the application handler. Deciding which parent to choose from the available SAP handlers depends on the actual use case of the extension. In some cases, it might even make sense to choose one of the SAP predefined object handlers as a parent.

5.3.1. Abstract Foundation Handler

Class CL_MDG_BS_FND_HANDLER is the abstract foundation handler in the MDG_FND software layer.

It contains generic and reusable coding for all entity types of MDG-BP, MDG-C and MDG-S. It introduces several static attributes and constants that are reusable by all other classes inheriting from the abstract handler. Some examples are:

- The MDG-BP specific derivation buffer GS_BP_EXTERN_DERIVED_DATA
- The MDG-BP specific active data buffer GT_BP_DATA_DB

5.3.2. Abstract Application Handler

Class CL_MDG_BS_ECC_HANDLER is the abstract application handler in the MDG_APPL software layer.

It contains only reusable coding for multiple assignments, MDG-C and MDG-S. It introduces several static attributes and constants that are reusable by all other classes inheriting from the abstract handler. Some examples are:

- The derivation buffer GS_MLT_AS_DERIVED_DATA
- The derivation task buffers BP_ROOT_TASK or GT_MLTAS_TASK
- The active data buffer GT_ECC_EXTERN_DB

5.3.3. Business Partner Handler

The MDG-BP specific handler class is CL_MDG_BS_BP_HANDLER.

It inherits from class CL_MDG_BS_FND_HANDLER. It provides the business partner-specific implementation of the handler interface.

This class is a valid parent for a custom handler class if the use case is a modification or enhancement of the existing application logic of the business partner.

If you have enhanced the data model with a custom entity type next to or directly below the business partner entity type, it’s recommended to create a custom handler class inheriting from the abstract foundation handler.

5.3.4. Multiple Assignment Handler

The multiple assignment specific handler class is CL_MDG_BS_MLT_ASSGNMNT_HANDLER.

It inherits from class CL_MDG_BS_ECC_HANDLER. It provides the multiple assignment-specific implementation of the handler interface.

This class is valid parent for a custom handler class if the use case is a modification or enhancement of the existing application logic of the multiple assignments.

If you have enhanced the data model with a custom entity type next to or directly below the multiple assignment entity type, it's recommended to create a custom handler class inheriting from the abstract application handler.

5.3.5. Customer Handler

The MDG-C specific handler class is CL_MDG_BS_CUST_HANDLER.

It inherits from class CL_MDG_BS_ECC_HANDLER. It provides the customer specific implementation of the handler interface.

This class is valid parent for a custom handler class if the use case is a modification or enhancement of the existing application logic of the customer.

If you have enhanced the data model with a custom entity type next to or directly below the customer entity type, it's recommended to create a custom handler class inheriting from the abstract application handler.

5.3.6. Supplier Handler

The MDGS specific handler class is CL_MDG_BS_SUPPL_HANDLER.

It inherits from class CL_MDG_BS_ECC_HANDLER. It provides the supplier (vendor) specific implementation of the handler interface.

This class is valid parent for a custom handler class if the use case is a modification or enhancement of the existing application logic of the supplier (vendor).

If you have enhanced the data model with a custom entity type next to or directly below the supplier entity type, it's recommended to create a custom handler class inheriting from the abstract application handler.

5.3.7. Data Derivation

The data derivation is part of the SAP handler classes. You can use data derivation to create, change, or delete data based upon either user actions in the UI or data inbound or both. An example is the automated creation of default partner functions as soon as a sales area is created in the customer UI. From the UI perspective, a derivation is triggered during the start of the UI application as well as during each UI round-trip triggered by the user. During data inbound derivations are executed, too.

The SAP handlers implement the derivation in two phases.

- The handlers analyze the current changes applied in the User Interface or in inbound processing. The application of these changes comes via the framework to the handler class. The data is buffered within the handlers. If any actions are required due to the new data, related tasks are being buffered, too. Related coding is available in the method BUFFER_DERIVED_DATA of the handler classes.
- The handlers perform the derivation according to the given tasks and buffered data. Related coding is available in the method DERIVE_DATA of the handler classes.

There is another special derivation option that is triggered by a change of the business partner key. In that case the MDG framework calls the handler method DERIVE_DATA_ON_KEY_CHANGE.

5.4. User Interfaces

The user interfaces (UI) for single object maintenance in MDG-BP, MDG-C and MDG-S are built using Floorplan Manager (FPM) and its Business Object Layer (BOL) / Generic Integration Layer (genIL) integration.

5.4.1. Generic Interaction Layer (genIL)

The genIL layer consists of a genIL model and one or more genIL implementation classes for the specific model.

genIL models are defined and maintained in the SAP backend with transaction GENIL_MODEL_BROWSER. genIL implementation classes are built within the common transactions SE24 or SE80.

The genIL model of MDG-BP, MDG-C and MDG-S reflects the introduced separation of objects due to multiple software layers, too. The basis is the business partner genIL model component BUPA which is extended by customer and supplier nodes in the genIL model enhancement BUPA_CUSP.

A genIL model basically consists of objects and relations.

- Objects consist of attributes. Each attribute reflects a usable field for the user interface.
- Relations connect one object to another. They define the cardinality of objects in a relation, too. Relations are reflected in the user interface by the wires (connections) from one UIBB to another. It is mandatory that the UIBB hierarchy in the overview page is consistent to the genIL object hierarchy as defined by the relations.

The genIL component BUPA is modelled. It is not generic. Any enhancement must be added manually. If you compare the genIL model BUPA (or BUPA_CUSP) with the MDG data model BP, you will notice that the hierarchical structuring is identical. It is important to add your enhancements at the same place. You will also notice that names differ. A mapping can be viewed for SAP defined entity type in view cluster VC_MDG_BS_GENIL respectively maintained for custom entity types in view cluster VC_MDG_BS_GENIL_C.

The important parts of an object are the key structure and the attribute structure. Both must be existing ABAP DDIC structures. The key structure is used internally by genIL. The sub-object's key structure must always contain the key fields of its parent. The attribute structure is used by FPM during the UI creation. It defines the field catalogue that is available for creating a list or form UIBB. If a key field shall be visible respectively maintainable in the UI, it must be part of the attribute structure, too.

Each genIL model component respectively enhancement requires exactly one implementation class. With regards to the complexity of the MDG-BP, MDG-C and MDG-S solution this might be considered as a limitation since the MDG solutions would need implementation classes for each specific object. To solve this challenge the MDG-BP, MDG-C and MDG-S genIL implementation consists of a strict class hierarchy where each class inherits from its parent class:

1. CL_MDG_BS_GENIL: the generic genIL implementation for all MDG solutions

2. CL_BS_GENIL_BUPA: the genIL implementation for MDG-BP

3. CL_BS_GENIL_MLT_ASSIGNMENTS: the genIL implementation for multiple assignments

4. CL_BS_GENIL_SUPPLIER: the genIL implementation for MDG-S

5. CL_BS_GENIL_CUSTOMER: the genIL implementation for all MDG-C

Class CL_BS_GENIL_CUSTOMER is defined in the genIL model enhancement BUPA_CUSP. The class hierarchy ensures that always all classes are executed during design and runtime of the UI. It is strictly mandatory that each custom enhancement of the genIL model which needs to use an own implementation class inherits from CL_BS_GENIL_CUSTOMER. Otherwise it might come to unforeseen errors during design and runtime of the UI.

The generic implementation in class CL_MDG_BS_GENIL contains a type-based mapping from the UI data structures into the MDG entity data structures and vice versa. This mapping works fine if the used data type of the source and target field is identical. If this is not the case, or if multiple fields are using the same data type, the generic mapping fails

which might lead to disappearing values in the UI. To resolve this issue, it is possible to re-define one of the following methods:

- IF_BS_TYPECASTED_MAP_ASSISTANT~TARGET_FIELD_NAME
  - The method allows defining a strict field mapping based upon the source structure and field name to the target structure and field name.
- IF_BS_TYPECASTED_MAP_ASSISTANT~TYPE_ALTERNATIVE
  - The method allows defining an alternative data type for the generic type-based mapping based upon the source structure and field type to the target structure and field type.
- "Example" implementations are pre-defined by SAP in the above-mentioned object-specific classes.

The business partner hierarchy of data model BP uses a dedicated genIL component called MDGBPH. The model is dynamically generated according to the MDG data model BP. Its implementation class is CL_MDGBP_GENIL_ADAPTER_HRY.

5.4.2. Floorplan Manager

The user interfaces (UI) for single object maintenance in MDG-BP, MDG-C and MDG-S are built using Floorplan Manager (FPM) and its Business Object Layer (BOL) / Generic Integration Layer (genIL) integration.

Some advantages by using FPM mentioned are:

- Loose coupling of the UIs to the MDG specific processes
- High flexibility for the creation of the UIs. The huge number of fields to be displayed is split into small UI Building Blocks (UIBBs). UIBBs support lists, forms and special kinds like pop-ups, search input and search results.
- Possibility to create object specific UIs to create a common look and feel and/or a similarity of the MDG UI compared to the SAPGUI maintenance transactions
- Reuse of existing tables, structures and fields (including naming) during the UI creation

5.4.3. Context Based Adaptations

A context-based adaptation (CBA) is an FPM concept that allows changing the UI in a flexible way based upon given values (e.g. application parameters, user input, and others). A CBA consists of an Adaptation Schema that consists of one or more Adaptation Dimensions. Using both it is possible to create various adaptations of the UI (e.g. a different layout of an overview page (OVP), or additional/removed row actions within a list UIBB, and so on). It is possible to combine several dimensions to create a very specific adaptation.

The CBA concept is based on the common FPM event handling. It is possible to trigger one or more CBA events that are handled by FPM's event loop processing.

Refer to chapter Context Based Adaptations (CBA) in the FPM Cookbook on SDN for more details.

The layout of an OVP can only be changed with a CBA during the startup of the application. It is not possible to change the OVP (e.g. the sequence of UIBBs) using a CBA during UI roundtrips. CBAs can only change the layout of single UIBBs for each round-trip.

FPM's event handling concept allows triggering multiple CBA events during one UI round-trip. But FPM does not cumulate the dimension information given within each event. Consider a scenario with the following sequence of events:

- A first event triggers a CBA for dimension ACTION (for example by defining the value DELETE to trigger the Mark for Deletion UI),
- A second CBA event contains only some information for a different dimension without the dimension ACTION.

In this scenario, the CBA for the ACTION will not be NOT processed since its value is reset to initial

Still, there is a possibility to trigger complex CBAs described in the related use case Trigger my own CBA Event.

The SAP delivered UIs support and use CBAs. There is a predefined schema BP_ADAPTS having several dimensions that are actively used within the UIs. The actual UI that is being displayed to a user in the web browser is determined from various components of the UI configuration:

- Personalization
- Enhancements
- Context Based Adaptations
- Base Configuration

The general rule is that the personalization is the strongest component. This is best explained with an example.

The base configuration defines the overview page as a list of UIBBs. Since a user does not want to scroll, he or she creates a personalization of the page introducing a stacking of the UIBBs in tab-strips. A UI designer decides to create a context-based adaptation that sets a single UIBB to “hidden and excluded from event loop”. All users not having a personalization will not see this UIBB anymore. The user with the personalization set is unaffected by this change. This is because the UIBBs that are hidden and excluded from event loop still belong to the OVP. They can be added to the OVP using personalization. Since the user has created a personalization that shows the UIBB (the personalization was created before the CBA), the UIBB is still visible. To exclude the UIBB you must either reset personalization or delete the UIBB in the CBA.

5.4.4. UIBBs & OVPs

UIBBs define the actual form or list usable for data maintenance. UIBBs are grouped into Overview Pages (OVPs) to finally build the UIs for end users.

UIBBs can be reused multiple times in different OVPs. This is done by the different MDG solutions to ensure a stable look and feel of the UIs. For example, the UIBB for maintaining the standard address is the same technical UIBB being used by nearly all the following OVP configurations.

MDG-BP provides the OVP BS_OVP_BP as main UI for business partner governance. You can find all UIBBs and configurations in package MDG_BS_BP_BOLUI.

MDG-C provides several independent UI applications for maintaining customer master data. All applications re-use common look and feel of the business partner UI and either change or extend it specifically for the customer:

- BS_OVP_CU
  - Main UI for customer governance.
  - Reuses the overview page for the business partner UI.
  - Extends this page with the customer multiple assignments list. This list is the starting point for the maintenance of ERP customer master data.
  - The UI consists of several edit pages with forms and list User Interface Building Blocks (UIBB).
- BS_OVP_CU_VL
  - Referred to as the ERP Customer Like user interface.
  - Does not use multiple assignments.
  - Layout is synchronized, as far as possible, with the old SAP ERP GUI transactions used to maintain customers (XD0*).
- BS_OVP_CU_LCC
  - Lean Requestor user interface for customer master data. The configuration consists only of a very limited sub-set of attributes.
- BS_OVP_CU_ORG
  - Lean Organizational user interface for customer master data. The configuration is usable only for data maintenance or approval steps. It consists of object specific sub-sets like only company codes or only sales areas. The intended usage is a scenario using the sub-workflow that assigns work items for company codes or sales areas to specific user groups only.

All objects are available in package MDG_BS_ECC_CUSTOMER_BOLUI.

MDG-S provides several independent UI applications for maintaining supplier master data. All applications re-use common look and feel of the business partner UI and either change or extend it specifically for the supplier:

- BS_OVP_SP
  - Main UI for supplier governance.
  - Reuses the overview page for the business partner UI.
  - Extends this page with the vendor multiple assignments list. This list is the starting point for the maintenance of ERP vendor master data.
  - The UI consists of several edit pages with forms and list User Interface Building Blocks (UIBB).
- BS_OVP_SP_VL
  - Referred to as the ERP Vendor Like user interface.
  - Does not use multiple assignments.
  - Layout is synchronized, as far as possible, with the old SAP ERP GUI transactions used to maintain vendors (XK0*).
- BS_OVP_SP_LVC
  - Lean Requestor user interface for supplier master data. The configuration consists only of a very limited sub-set of attributes.
- BS_OVP_SP_ORG
  - Lean Organizational user interface for supplier master data. The configuration is usable only for data maintenance or approval steps. It consists of object specific sub-sets like only company codes or only purchasing organizations. The intended usage is a scenario using the sub-workflow that assigns work items for company codes or purchasing organizations to specific user groups only.

All objects are available in package MDG_BS_ECC_SUPPLIER_BOLUI.

The multiple assignment UI consists of re-usable components only. There is no specific OVP. There are several lists and forms available for the different assignment categories of business partner, customer and supplier. This simplifies the creation of object specific OVPs. All objects are available in package MDG_BS_ECC_BP_MLT_ASSGMNT.

If you use both MDG-C and MDG-S, OVP BS_OVP_BP_ALL combines both the customer and supplier related UIBBs into one, singe OVP.

www.sap.com/contactsap

THE BEST RUN SAP®