Enrichment of Master Data in MDG – Generic Guide and Sample Implementation

TABLE OF CONTENTS

Contents

INTRODUCTION

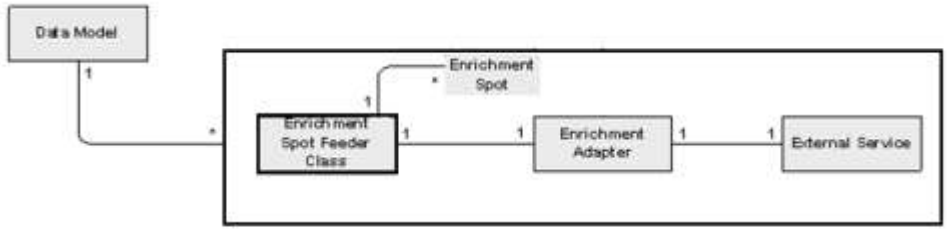
You can use the enrichment framework to enrich the MDG data with external services or with internal logic. The enrichment framework also supports embedding of specific UIs for enrichment. The first section of this guide provides a generic overview of how enrichment works. The second section provides an example of address validation.

SECTION A: ENRICHMENT GENERIC GUIDE

Support
CSS component: `CA-MDG-DQ-ENR` (*Cross Applications - Master Data Governance – Data Quality-Enrichment*)
Prerequisites
Development Skills
Knowledge of ABAP OO and Web Dynpro for ABAP
System Requirements
This document describes Enrichment Framework as it is available with Master Data Governance EhP6 which is part of the Software Component `SAP_BS_FND` release 7.31. In addition, the guide describes new features that are available as part of EhP6.1
Important Terms

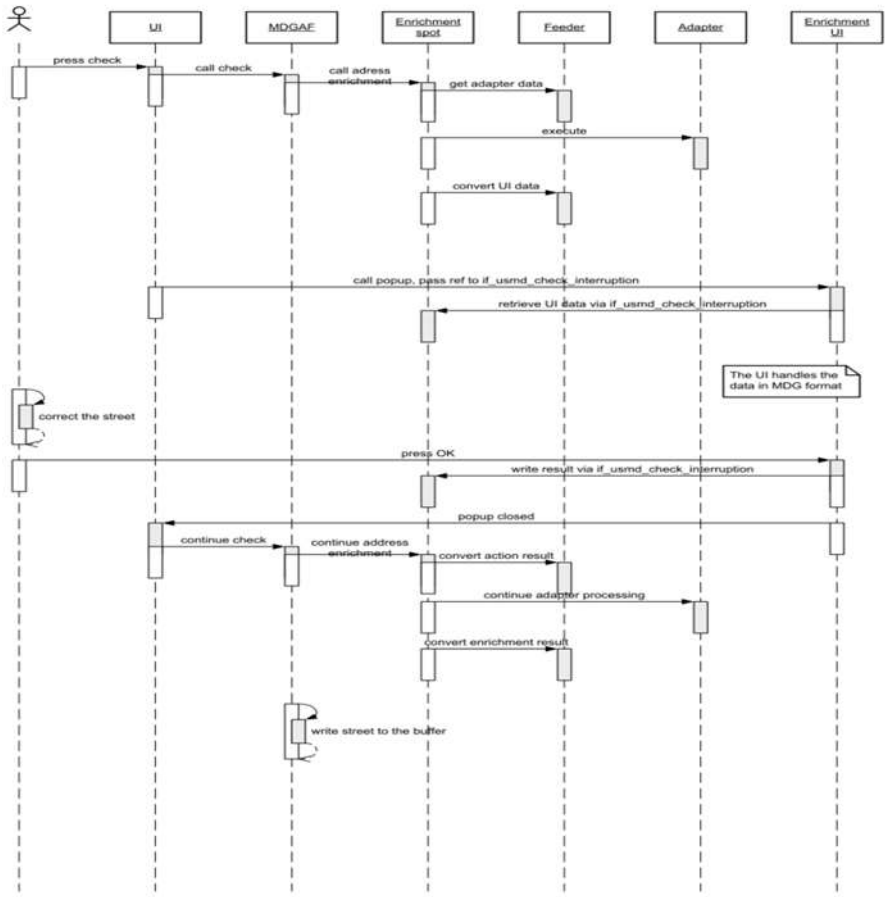
Term	Definition
Enrichment Spot	The place holder where an enrichment of certain data in MDG is possible. For example, the *Address Enrichment* enrichment spot supports the enrichment of addresses of BP/Supplier/Customer
Enrichment Adapter	An implementation of the interface `IF_USMD_ENRICHMENT_ADAPTER` for a particular enrichment. The adapter makes the call to the external/internal service to get the data necessary for enrichment.
Enrichment Feeder	An implementation of the interface `IF_USMD_ENRICHMENT_FEEDER` for a particular enrichment. A feeder is mainly used to convert data (for structure mapping and for data mapping) from the MDG format to the enrichment adapter format as well as from the enrichment adapter format to the MDG format.
Enrichment UI	Any enrichment can have its own UIs, which are shown during the process of enriching the MDG data in cases where user interaction is necessary. The UI must be a WebDynpro ABAP component and must implement the WebDynpro interface `MDG_ENRICHMENT_INTF`. **NOTE:** There can be only one adapter implementation for a third-party service and it can be mapped only to one enrichment spot and enrichment feeder.

Block Diagram and Sequence Diagram
The graphic below illustrates the broader level interaction between the various components:



A Data model can have 1:N enrichment spots. One enrichment spot must have a feeder class, an adapter class, and an enrichment UI component. The same feeder class can be associated with 1:N enrichment spots. Since an adapter is the component that interacts with the third-party service, it does not make sense to use the same adapter classes for various enrichments.

The interaction between the various components involved in enrichment can be seen in the sequence diagram below:



Methods and APIs Used to Create a Simple Enrichment

We recommend that you create your own ABAP structures for the following:
- Adapter input format – Based on the data needed by the adapter
- Adapter output format – Based on the data returned from the adapter
- MDG format for adapter output (to update the data into MDG)
- MDG format for adapter input (to retrieve the data from MDG)
- UI data to be displayed in MDG format

Enrichment Feeder
Feeder class implements the interface `IF_USMD_ENRICHMENT_FEEDER`. The interface has following methods:
Method: *`IF_USMD_ENRICHMENT_FEEDER~GET_RELEVANT_ENTITIES`*
This method is used to retrieve the information:
1) The relevant entities that has to be read
2) The relevant entity that has to be written onto.
3) The relevant attributes of the entity that has to be written onto (**new feature in MDG 6.1**) (These particular attributes have to be mandatorily editable or else the enrichment won't run. It doesn't make sense for enrichment to update data onto read-only fields)
4) These attributes of the entity that has to be written onto aren't mapped for enrichment, hence these attributes' values would be preserved after enrichment.

Parameter	Direction	Type Kind	Type	Description
ET_READ_ENTITIES	Exporting	Type	USMD_T_ENTITY	Entities that have to be read
E_WRITE_ENTITY	Exporting	Type	USMD_ENTITY	Entity that would be enriched
ET_ATTRIBUTE	Exporting	Type	USMD_TS_FIELDNAME	Attributes of the entity that has to be enriched
ET_ATTR_NO_MAP	Exporting	Type	USMD_TS_FIELDNAME	Attributes of write entity not mapped for enrichment

Example

DUNS enrichment reads `AD_POSTAL`, `BP_CENTRL` and `BP_ADDUSG` as input data to determine details of the entity type `BP_IDNUM`, which needs to be updated. The following attributes of `BP_IDNUM` are being updated and so must be editable: `BP_ID_TYPE` and `BP_ID_NUM`. The code relating to these entity types and attributes should look something like this:
`et_read_entities : "AD_POSTAL", "BP_CENTRL", "BP_ADDUSG"; e_write_entity : "BP_IDNUM"`
`et_attributes : "BP_ID_TYPE", "BP_ID_NUM; et_attr_no_map : "ID_VFROM", "ID_VTO".`

Method: *`IF_USMD_ENRICHMENT_FEEDER~GET_ADAPTER_DATA`*

This method retrieves the data necessary for the adapter (in the adapter format) to execute the enrichment. While `GET_RELEVANT_ENTITIES` is implemented, the framework takes care of extracting the data that has to be read.

Parameter	Direction	Type Kind	Type	Description
IO_APP_CONTEXT	Importing	Type Ref To	IF_USMD_APP_CONTEXT	Application context contains the CR type, CR ID, Business Activity

				chosen etc.
IO_MODEL	Importing	Type Ref To	IF_USMD_MODEL_EXT	Reference to the model extension interface that enables necessary data to be read from MDG
IT_KEY_ALL	Importing	Type	USMD_TS_ENTITY_DATA	All the keys and the data involved in the current CR (For example, in the case of supplier, the entities are CR, BP_HEADER and ADDRNO.)
ER_DATA	Exporting	Type Ref To	DATA	Data in adapter format

Sample code is shown below:

```
LOOP AT IT_KEY_ALL ASSIGNING <LS_KEY_ALL>.
ASSIGN <ls_key_all>-r_data->* TO <lt_data>.
 CASE <ls_key_all>-entity.
       WHEN 'xyz'
             LOOP AT <lt_data> ASSIGNING <ls_data>.
                   <Convert data to adapter format>
             ENDLOOP.
       .
       .
       WHEN 'nth case'
             LOOP AT <lt_data> ASSIGNING <ls_data>.
                   <Convert data to adapter format>
             ENDLOOP.
 ENDCASE.
ENDLOOP.
```

Method: IF_USMD_ENRICHMENT_FEEDER~GET_MDG_DATA

This method converts the data from adapter format to MDG format. The data which is in MDG format is sent back to the framework to be written onto.

Parameter	Direction	Type Kind	Type	Description
IT_DATA	Importing	Type	STANDARD TABLE	Table of enriched data (in adapter format) that can be written back to MDG. This structure contains the following components: • WRITE_DATA contains the enriched data in adapter format • ATTRIBUTE_LIST contains the list of attributes that have changed after the enrichment. The attribute list contains the fields from the adapter structure.
IO_DATA_WRITE	Importing	Type Ref To	IF_USMD_DELTA_BUFFER_WRITE	Buffer instance that can be used to write data into MDG using it's WRITE_DATA method (in cases where the writing logic is to be handled in the feeder implementation)

ER_DATA	Exporting	Type Ref To	DATA	Data in MDG format

Sample steps in this method would look as follows:

```
Loop at IT_DATA assigning <ls_data>
 Convert <ls_data>-write_data into MDG format
 Convert <ls_data>-attribute_list into MDG fields
End Loop.
```

Method: IF_USMD_ENRICHMENT_FEEDER~PREPARE_UI_DATA

This method is used to provide the data to the UI by converting it from adapter format to MDG format. The view type indicates the view that is currently displayed. In EhP6, the view type is hardcoded to SCR1 which means it is the initial screen. The data returned in ER_DATA from this method is sent to the WebDynpro UI component using the SHOW_ENRICHEMENT_UI method.

Parameter	*Direction*	*Type Kind*	*Type*	*Description*
IV_VIEW_TYPE	Importing	Type	CHAR4	View Type (hardcoded as SCR1)
IR_DATA	Importing	Type Ref To	DATA	Data necessary for UI interaction returned by the adapter in adapter format
ER_DATA	Exporting	Type Ref To	DATA	Data in MDG format for UI

Method: IF_USMD_ENRICHMENT_FEEDER~CONVERT_ACTION_RESULT

This method is used to convert the UI result from the MDG format to the adapter format for further processing by the adapter. The UI result is in the importing parameter IR_DATA which is received from the event END_UI_INTERACTION of the WebDynpro component (see Step 3 for more details).

For example: While executing an enrichment, a user interaction is necessary and a corresponding UI is shown to the user with a list of values (in MDG format) to choose from. The user chooses a row from the list and clicks on OK. This value needs to be returned to the adapter for processing. Before passing the data chosen on the UI to the adapter, this method is called to convert the format from MDG format to adapter format.

Parameter	Direction	Type Kind	Type	Description
IV_ACTION_CODE	Importing	Type	MDG_SDQ_ENRICH_ACTION_CODE	Action Code
IR_DATA	Importing	Type Ref To	DATA	Reference to UI data returned with action code
ER_DATA	Exporting	Type	DATA	Return data in adapter format

Method: IF_USMD_ENRICHMENT_FEEDER~IS_RELEVANT
This method is implemented to specify whether the current enrichment makes sense for the current OTC code (business object) being dealt with in the CR.

This method checks the customizing for the following:

- Whether or not, the enrichment has been made relevant
- Whether the enrichment is relevant for the particular data model.

Parameter	Direction	Type Kind	Type	Description
IV_OTC	Importing	Type	USMD_OTC	Business Object Type
EV_RELEVANT	Exporting	Type	BOOLE_D	Whether this enrichment is relevant for a BO Type or not

Enrichment Adapter
Create a class which implements the interface IF_USMD_ENRICHMENT_ADAPTER. The interface has following methods:

Method: IF_USMD_ENRICHMENT_ADAPTER~EXECUTE

This method makes the call to the external service or the internal service to get the enriched data. In case a user interaction is necessary to complete the enrichment, the method sets the EF_USER_ACTION_NECESSARY flag as well as sending the data to be enriched in the ER_DATA parameter.

If data enrichment is unnecessary, the EF_USER_ACTION_NECESSARY flag is set to false, along with and the method sends back the data as being perfectly validated.

Parameter	Direction	Type Kind	Type	Description
C_LAST_USER_ACTION	Changing	Type	MDG_SDQ_ENRICH_ACTION_CODE	Action Code
ET_MESSAGE	Exporting	Type	USMD_T_MESSAGE	Messages
EF_USER_ACTION_NECESSARY	Exporting	Type	USMD_FLG	Indicator to specify whether an user interaction is necessary or not
ER_DATA	Exporting	Type Ref To	DATA	Data that is needed for displaying the UI if user interaction is necessary
ET_ADDITIONAL_DATA	Exporting	Type	USMD_TS_VALUE	Name value pair to hold any additional data necessary for UI interaction
CS_DATA	Changing	Type Ref To	DATA	Data to be enriched

UI component for User Decision Dialogs

If there is a user interaction expected for the enrichment to be developed, a WebDynpro component with the necessary UIs must be created. The WebDynpro component must implement the WebDynpro component interface `MDG_ENRICHMENT_INTF`. This interface uses method `SHOW_ENRICHMENT_UI`.

Method `SHOW_ENRICHMENT_UI`:

This method is used to start the enrichment UI.

Parameter	Direction	Ref To	Opt	Type	Description
IR_DATA	Importing	Ref To		DATA	Data relevant for UI interaction in MDG format that was sent by adapter in the execute method
IR_WINDOW	Importing	Ref To		IF_WD_WINDOW	Window instance that can be used to embed the enrichment UIs
IT_ADDITIONAL_DATA	Importing		Opt	USMD_TS_VALUE	Additional data needed for UI interaction sent by adapter in the execute method.

Event: `END_UI_INTERACTION`

This event must be started to indicate the end of the user interaction so that the enrichment can continue with the provided UI data.

Parameter	Ref To	Opt	Type	Description
ER_DATA	Ref To		DATA	The data entered on the UI by the user after the UI interaction
EV_ACTION_CODE			CHAR4	The action code based on the action the user has chosen.

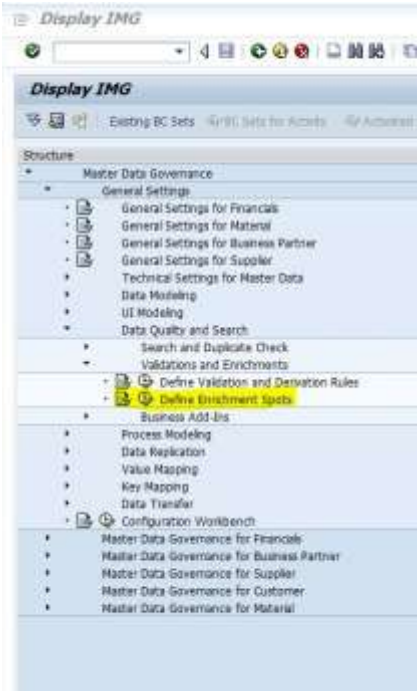
The following action codes must be set to indicate the completion of the UI interaction:

- `ENRI`: Use the enriched data and complete the interaction
- `IGNR`: Ignores the current enrichment and perform it again in the next check cycle
- `ORIG`: Discard the enriched data and continue with the data entered on the UI by the user
- `CNCL`: Stops the check cycle and the control goes to the UI.

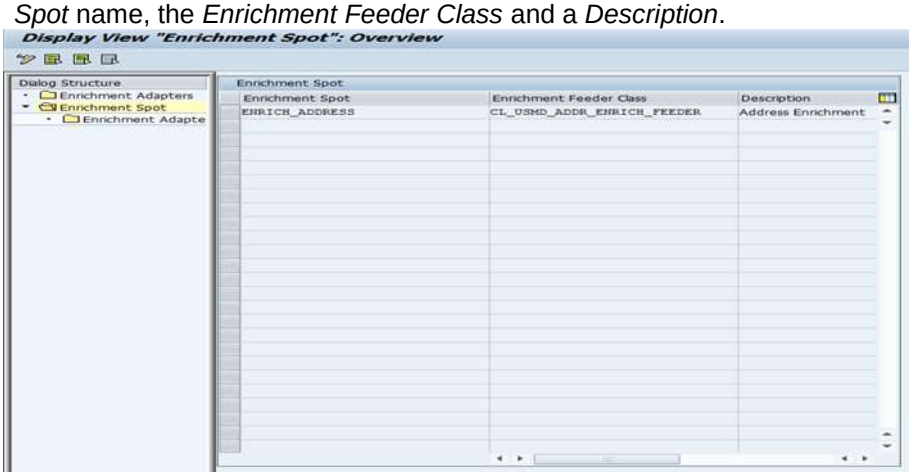
Definition of an Enrichment Spot

After you create the above objects, you can perform the necessary customizing to execute the enrichment.

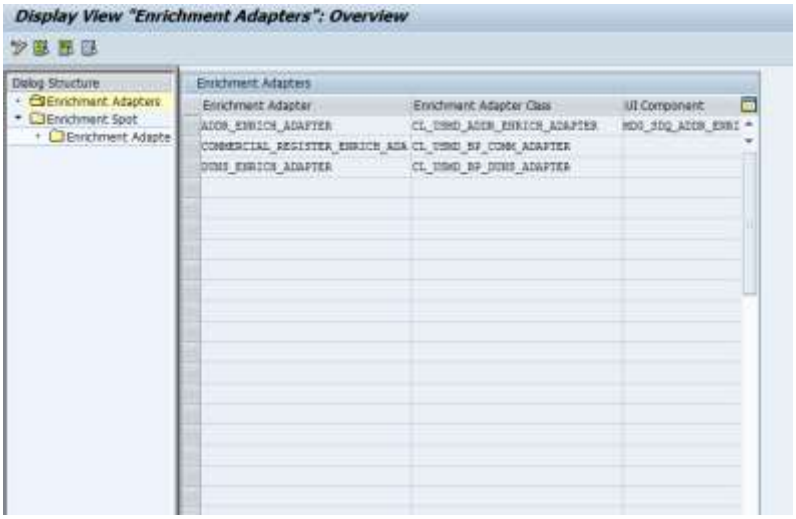
Run the *Define Enrichment Spots* activity in Customizing for *Master Data Governance* (transaction `MDGIMG`) as shown in the following screenshot:



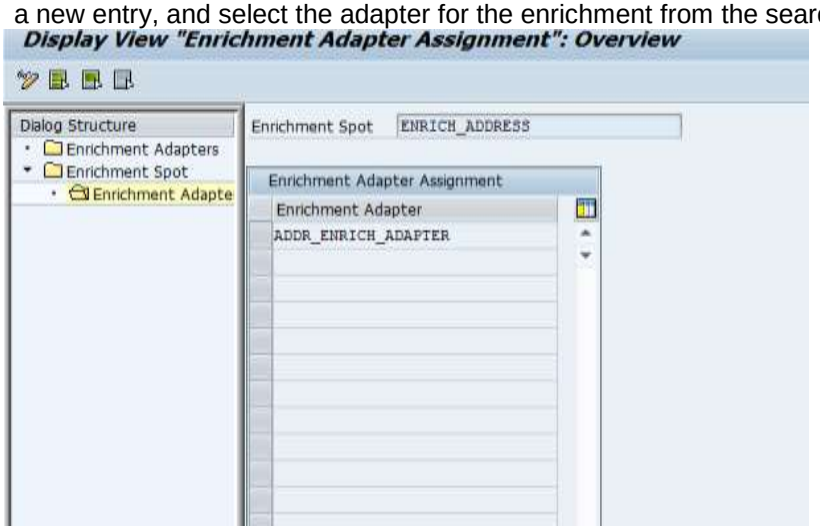
1. Select the *Enrichment Spot* folder, create a new entry, and enter necessary details such as *Enrichment Spot* name, the *Enrichment Feeder Class* and a *Description*.



2. Select the *Enrichment Adapters* folder, create a new entry, and enter the necessary details such as *Enrichment Adapter* name, *Enrichment Adapter Class*, the WebDynpro UI component for the enrichment, and a description for the adapter.

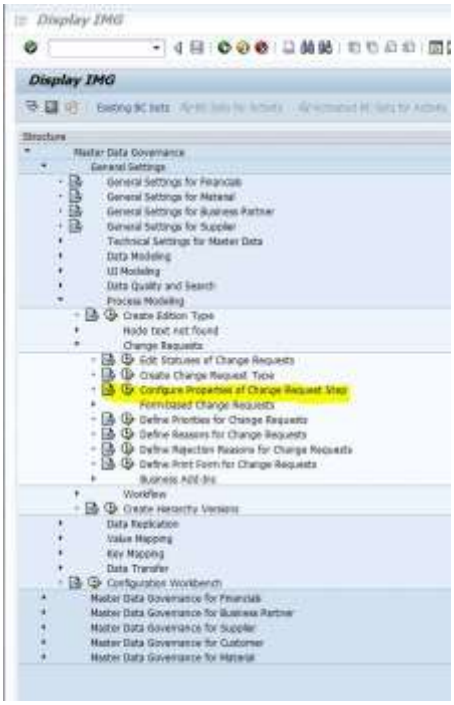


3. Select the *Enrichment Spot* created above and select the node *Enrichment Adapter Assignment*, create a new entry, and select the adapter for the enrichment from the search help.



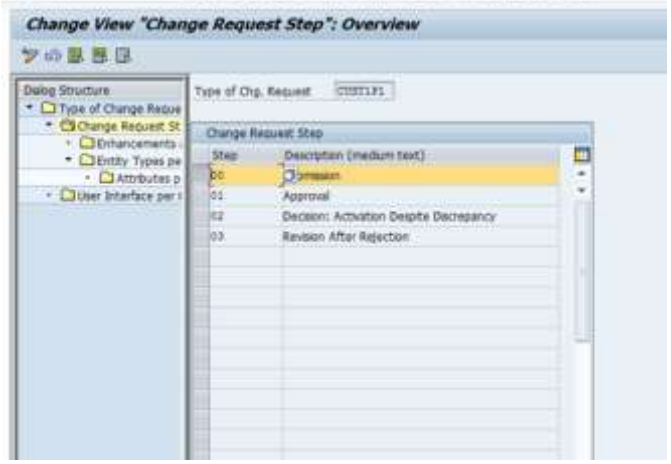
Configuration Required to Ensure Enrichment Occurs at Specific Change Request Steps

1. Configure the change request step properties for the execution of enrichment during one or more change request steps. To ensure that the enrichment gets executed at the necessary steps within the change request, run the customizing activity *Configure Properties of Change Request Step* in Customizing for *Master Data Governance* (transaction `MDGIMG`) as shown in the following screenshot:

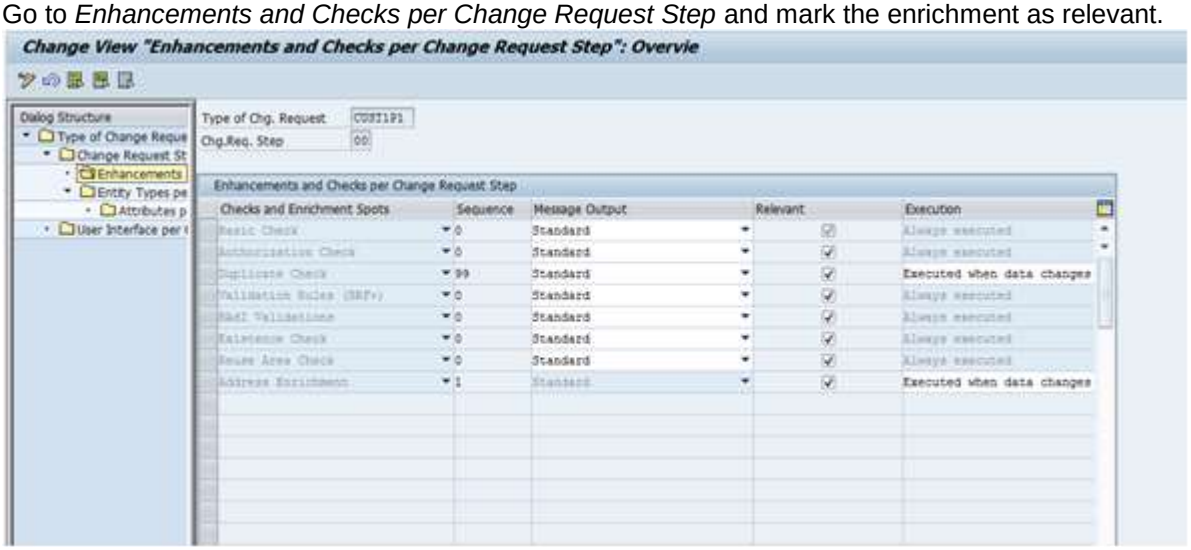


2. Select the *Type of Change Request* for which you wish to configure the enrichment.

3. In the *Change Request Step* view, select the *Change Request Step* for the type of change request.



4. Go to *Enhancements and Checks per Change Request Step* and mark the enrichment as relevant.



SECTION B: SAMPLE IMPLEMENTATION: ADDRESS VALIDATION

Steps to integrate any third party solution for address validation into MDG:

1. Implement BAdI `ADDRESS_CHECK`.
2. Create a third party WebDynpro component.
3. Create UI Component for user decision dialogs.
   Complete reuse is possible for this step.
4. Maintain error codes.
   The appropriate view is `V_MDG_SDQ_MSG_UI`.
5. Create a feeder class
6. Create an adapter class for the following scenarios:
   - 1:1 mapping scenario
   - Addresses with error

   Complete reuse is possible for this step.

Detailed Steps

Implement BADI `ADDRESS_CHECK`

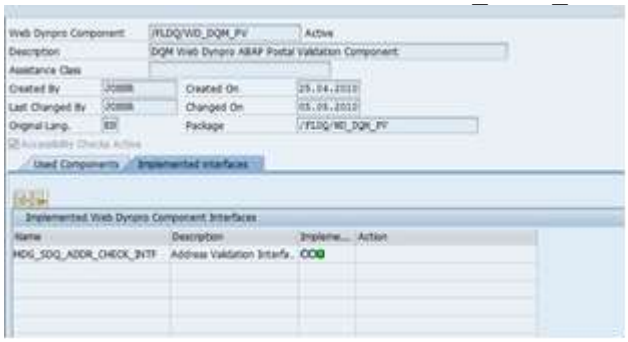
It is expected that the third party service implements the standard BAdI provided for validation: `ADDRESS_CHECK`.

Create Third Party WebDynpro Component

There are two scenarios possible:
1. The address validation-3rd party service already uses WD ABAP technology
2. The address validation-3rd party service does not use ABAP technology.

For scenario 2, you must first create the corresponding implementation in WebDynpro ABAP.

Note: The WebDynpro component must implement the interface: `MDG_SDQ_ADDR_CHECK_INTF`. When the user interaction is over and the `NOTIFY_INTF_COMP` event has to be triggered.



Create UI Component for User Decision Dialogs

(Example `ZMDG_SDQ_ADDR_ENRICHMENT`)

Interface to be implemented: `MDG_ENRICHMENT_INTF`

If a user interaction is expected for the enrichment to be developed, you must create a WebDynpro component with the necessary UIs. The WebDynpro component must implement the WebDynpro component interface `MDG_ENRICHMENT_INTF`.

Now in the `EXECUTE` method of the adapter class, the additional details, such as the UI component details are also completed. Before the specific implementation `SHOW_ENRICHMENT_UI` is triggered from the generic implementation `SHOW_ENRICHMENT_UI` of generic WebDynpro component, `MDG_ENRICHMENT`, `get_ui_data` is triggered. `get_ui_data` picks up the additional details from the adapter implementation using the execute method.

For details of events SHOW_ENRICHMENT_UI and END_USER_INTERACTION, refer to the generic guide.

Note

The MDG_SDQ_ADDR_ENRICHMENT component can be re-used entirely if the third-party WebDynpro component implements: MDG_SDQ_ADDR_CHECK_INTF.

Make sure that adapter~execute implementation fills the additional details in the global parameters if the UI component for user decision dialog and the prepare_ui_data feeder method are re-used (but not necessarily the private method fetch_enrich_data).

Maintain Table MDG_SDQ_MSG_UI

View: V_MDG_SDQ_MSG_UI

Example: FLDQ Component

Table: MDG_SDQ_MSG_UI
Displayed Fields: 7 of 7 Fixed Columns: 3 List Width 0250

CLIENT	ARBGB	MSGNR	MSGTY	SUPPRESS	POPUP_COMP_NAME	TITLE_OTR_ALIAS
405	/FLDQ/AD_ADDR_ERRORS	000	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	001	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	002	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	020	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	101	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	102	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	103	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	104	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	105	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	212	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	213	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	214	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	216	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION
405	/FLDQ/AD_ADDR_ERRORS	302	E		/FLDQ/WD_DQM_PV	/FLDQ/WD_DQM_PV/ADDRESS_VALIDATION

If you have to reuse the adapter~execute method, you must maintain the table shown above. Based on the implementation of the BAdI method: address_postal_check, the method may return an error code: MSGNR. The error code returned, consisting of POPUP_COMP_NAME and TITLE_OTR_ALIAS is filled in global parameters for additional details.

You can disable suggestion lists, preventing them from being displayed for specific error codes, by selecting the suppress option for the error code. This feature is currently handled in the execute method of the standard adapter class.

Create Feeder Class

Example Table: ZCL_USMD_ADDR_ENRICH_FEEDER

Interface to be implemented:

IF_USMD_ENRICHMENT_FEEDER

Methods that can be re-used:

- GET_RELEVANT_ENTITIES: Gives the entity name that has to be read and written onto. In the case of address enrichment, the same entity is read and updated. For example, AD_POSTAL.
- IF_RELEVANT: This method is checks whether the current enrichment makes sense for the Object Type Code (OTC) of the business object being dealt with in the CR. In this method, the system checks whether the OTC is *147* or not. 147 is the OTC for the Business Partner data model.
- GET_ADAPTER_DATA : This method retrieves the data necessary for the adapter (in the adapter format) to execute the enrichment. Note: You might have to modify the private method fetch_adapter_data, which is called in this method when structures are not reused.
- GET_MDG_DATA : This method is used to convert the data from adapter format to MDG format. Note: Note: You might have to modify the private method fetch_mdg_data, which is called in this method when structures are not reused.

Methods that have to be modified:

As mentioned in the recommendations, it is advisable to create your own ABAP structures for the following:

- The adapter input format, which is based on the data needed by the adapter.

- The adapter output format, which is based on the data returned from the adapter.
- The MDG format for the adapter output, which is used to update the data into MDG.
- The MDG format for adapter input, which is used to retrieve the data from MDG.

The methods are as follows:

- `FETCH_ADAPTER_DATA`
  The data sent from `GET_ADAPTER_DATA` is in MDG format. The same code in the method can be used, provided the structure mentioned in the code can be reused for the third-party service being used.
- `FETCH_MDG_DATA`
  The data sent from `GET_ADAPTER_DATA` is in adapter format. It has to be converted to MDG format before the entity gets updated. The same code in the method can be used, provided the structure mentioned in the code can be reused for the third-party service being used.
- `PREPARE_UI_DATA`
  This method is used to provide the data to the UI by converting it from adapter format to MDG format. It is triggered from `GET_UI_DATA`, which in turn is triggered from the `SHOW_ENRICHMENT_UI` of the generic WebDynpro component `MDG_ENRICHMENT`. The following use cases are possible:
  - The user decision dialogs are handled by the third-party service.
    In this case, the data sent to the `PREPARE_UI_DATA` method - which is in an adapter format that is recognizable by the third party service - can be returned as it is. In this case, the method can be re-used, along with the private method `fetch_enrich_data`.
  - The display is to be MDG format.
    In this case, the `prepare_ui_data` method can be re-used. However, `fetch_enrich_data` must handle the conversion from adapter format to MDG format and must be modified accordingly.

- `CONVERT_ACTION_RESULT`
  This method is used to convert the UI result which is in MDG format to adapter format for further processing by the adapter. The UI result is represented in the importing parameter `IR_DATA`, which is received from the event `END_UI_INTERACTION` of the WebDynpro component.

  Example
  While executing an enrichment, a user interaction is necessary. A corresponding UI presents the user with a list of values (in MDG format). The user chooses a row from the list and chooses *OK*. This value needs to be returned to the adapter for processing. Before passing the data chosen on the UI to the adapter, the `CONVERT_ACTION_RESULT` method converts the data from MDG format to adapter format.

  If the display is handled by the third party service, the data returned from the UI is in adapter format itself and the importing data can be sent back as exporting data as well. In this case, the method can be re-used.
- `FETCH_ADAPTER_DATA`
  This method is a private method in `GET_ADAPTER_DATA`. This method should do the conversion from MDG format to a format that the third-party service recognizes (in other words, the adapter format).
- `FETCH_MDG_DATA`
  This method is a private method in `GET_MDG_DATA`. This method does the conversion from format that the third-party service recognizes (the adapter format) to the MDG format.

Create Adapter Class

An example class is `ZCL_USMD_ADDR_ENRICH_ADAPTER`

Interface to be Implemented
`IF_USMD_ENRICHMENT_ADAPTER`

Method that Can Be Reused

`EXECUTE`: This method makes the call to the external or internal service to get the enriched data. In case a user interaction is necessary to complete the enrichment, the flag `EF_USER_ACTION_NECESSARY` is set along with sending the data `(ER_DATA`) necessary to be displayed on the UI

.

1:1 Mapping Scenario

For addresses that only contain a slight error, the popup is suppressed, no error code is sent and the address is immediately corrected without any user intervention.

For addresses that require user-intervention for validation: The initial popup to be displayed is being determined in this method by using the error-code sent by the check_postal_address method and subsequently pick up the additional details from the table MDG_SDQ_MSG_UI based on the error code sent. Refer to Maintenance of table MDG_SDQ_MSG_UI.

www.sap.com