PUBLIC

How-To: Enable system as application data owner in classic mode for federated processes

SAP Master Data Governance

Applicable Releases:

From S/4HANA 2023 FPS01 for core data owner

From S/4HANA 1809 for application data owner

Version 1.0

February 2024

Document History

Document Version	Description
1.0	First official release of this guide

1 Business Scenario

As of SAP S/4HANA 2023 the new cloud-ready mode of SAP Master Data Governance is available. New SAP Fiori user interfaces are provided with this new mode. This cloud-ready mode must be switched on to enable the new features.

One of these new features is federation in SAP Master Governance. The prerequisite for federation is that all participating systems run on SAP S/4HANA 2023 or higher.

As of SAP S/4HANA 2023 FPS01 there will be a new option for customers, who cannot update all participating application data owners (ADO) to the required SAP S/4HANA release. They can use systems running on lower SAP S/4HANA releases by setting up classic central governance processes (based on MDG change requests). These systems can act as application data owners in federation processes.

But the system, which acts as core data owner (CDO) must run on SAP S/4HANA 2023 FPS01 or higher.

Besides the configuration of the core data owner system, which is described in the configuration guide, there are several custom enhancements for the application data owners that are required to get federation running. This how-to guide outlines all the required configurations and enhancements.

2 Configuration and Enhancements for Application Data Owners in classic mode

The following configurations and enhancements must be implemented on all affected application data owner systems, to involve them in federation processes.

2.1 Federation Customizing

The general configuration for federated processes should be executed before the extensions on application data owners will be implemented. Please check the required settings in the following documentation:

Specify Application Data Owners in Classic Mode

2.2 MDG, Central Governance Customizing

2.2.1 Create Change Request Types

Perform the following steps in the application data owner system in classic mode:

1. Start transaction MDGIMG and execute the IMG activity Master Data Governance, Central Governance--> General Settings--> Process Modeling--> Change Requests--> Create Change Request Type.

2. Create change request type(s) to be used for inbound processing. The change request creation will be triggered by inbound SOAP messages coming from the core data owner system.

3. Depending on your requirements you can assign the following business activities to the change request:
   - BPPI (MDG Hub Inbound: Business Partner)
   - BPPU (MDG Hub Inbound: Business Partner)
   - BPPL (Business Partner Initial Load)

You can refer to pre-delivered change request types CUSTHI01, CUSTHI02, BPLP1.

4. Maintain agent determination for the change requests according to the needs of your organization.

2.2.2 Configure Properties of Change Request Step

Perform the following steps in the application data owner system in classic mode:

1. Start transaction MDGIMG and execute IMG activity Master Data Governance, Central Governance--> General Settings--> Process Modeling--> Change Requests--> Configure Properties of Change Request Step.

2. Select your change request type and double-click on ‘Change Request Step’ in the dialog structure. Choose step 00 and double-click on ‘Enhancements and Checks per Change Request Step’. Remove the ‘Relevant’ flag for all checks apart from ‘Basic Check’.

3. In the dialog structure choose ‘Entity Types per Change Request Step’. For all entity types select field properties ‘No Required Field Check’.

4. Navigate one step back, choose a step for manual processing of change request data (for example step 02 ‘Approval (No Rejection)’ in the pre-delivered change request type CUSTHI01, step 01 ‘Approval (No Rejection)’ in pre-delivered change request type CUSTHI02 or step 01 ‘Processing’ in pre-delivered change request type BPLP1) and double-click on ‘Entity Types per Change Request Step’ in the dialog structure.

5. Maintain field properties for the entity types according to your defined data ownership model. For the entity types for which the ADO in classic mode is defined as a data owner choose ‘Standard’. For all other entity types choose ‘Not Relevant’. Save your entries.

Note:

It’s not possible to protect filtered entities such as single company codes, sales areas, or identification numbers of certain types.

2.3 DRF Customizing Settings

Perform the following steps in the **application data owner system in classic mode:**

1. Start transaction DRFIMG and execute IMG activity Define Custom Settings for Data Replication → Define Technical Settings → Define Technical Settings for Business Systems.

2. Mark/add the entry for Core Data Owner Business System and navigate to ‘Define Bus. Systems, BOs’. Choose/add BP Type ‘147’ and mark the entry, choose ‘Define Bus. Systems, BOs, Communication Channel’:

3. The Communication Channel should be ‘Replication via Services’, Key Harmonization is ‘Key Mapping’. Storage should be “Active Area, but in case of error Staging Area”.

Note:

This setting ensures that BPs sent in a SOAP message from federated processes will not be activated directly but instead written into the staging area (in an MDG change request).

Furthermore, Core Data Owner needs to be added to DRF configuration for outbound replication:

1. Start transaction DRFIMG and execute IMG activity Define Custom Settings for Data Replication → Define Replication Models

2. Choose Replication model for Replication of BP and BPRel via SOA (Assign Outbound Implementation has Outbound Implementation ‘986_3’) and add Core Data Owner Business System in 'AssignTarget Systems for Repl. Model /Outb.Impl’

2.4 MDG, Central Governance Enhancement implementation

As already described in the first section of this guide, an MDG, Central Governance system on SAP S/4 HANA 1809 or higher can act as an application data owner within an MDG Federation process. To do this, several BAdIs and Enhancement Implementations are necessary.

Note that all mentioned lines of source code are examples only and can be adjusted or removed any time by SAP. Any implementation of any kind of enhancements or modification are subjects for Custom Implementation Projects! Source code examples can be found in the section “SAP Source Code Examples”.

The goal is to keep the necessary custom implementations as small as possible so that an SAP S/4HANA On-Premise system with MDG, Central Governance can act as an Application Data Owner in Classic Mode (ADOCM) within a Federation process. Whenever the Central Data Owner (CDO) system sends a Business Partner Bulk Replication Message which relates to a Federation process, the BusinessScope entity of the singe message Header is filled. Based on this data, ADOCM systems can react to such messages and can create MDG, Central Governance Change Requests.

2.4.1 Implementation for BAdI MDG_SE_BP_BULK_REPLRQ_IN

The single object message header contains the information that notes if the incoming message is related to a Federation process. This information is stored in the entity BusinessScope.

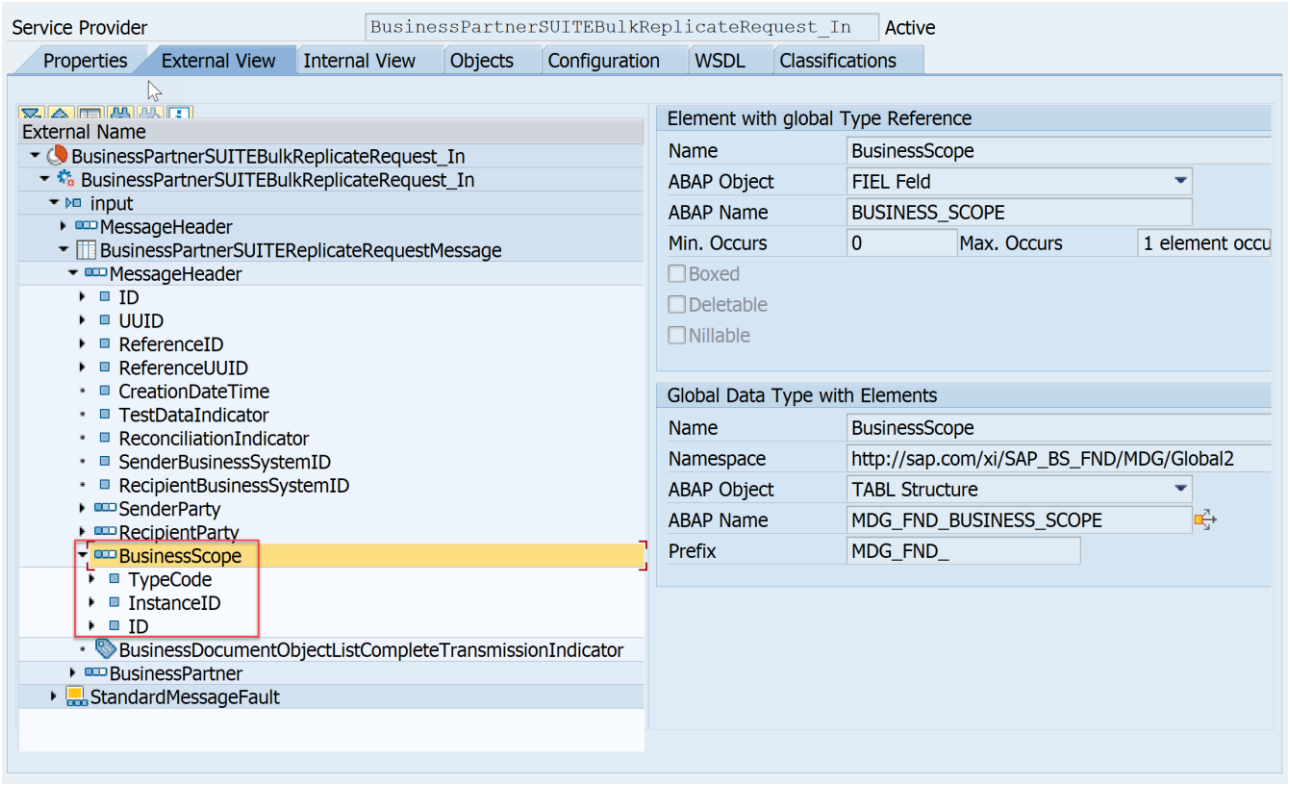

Figure 1: Business Scope in single message header

The Business Scope contains the following elements:

- TypeCode
  Contains the action, which the federation process expects to be done by ADO. ADOCM supports for actions and treats them all in the same way: it tries to create a new MDG, Central Governance Change Request. Supported actions are:
  - APP – Approve
  - APA – Approve again
  - MRG – Merge
  - MRA – Merge again
- InstanceID
  Contains the federation process ID on CDO system, the current federation step number, and the size of the match group; 1 always means that the match group consists of only one record and no duplicates.
- ID
  The ID of the best record of the match group on CDO system. In the case of TypeCode APP or APA it's always the UUID of the Business Partner of the BusinessPartner-entity.
  Note that to ADO systems in classic mode no duplicates are sent but only the best record.

This Business Scope is required in several places within the inbound processing logic. Therefore, it must be stored and be accessible in those places. To store the single message header, implement BAdI

MDG_SE_BP_BULK_REPLRQ_IN (Enhancement Spot MDG_SE_SPOT_BPBUPA) in your ADOCM system.

The implementing class should:

- Have a public class table attribute `CLASS-DATA messageHeaders TYPE SORTED TABLE OF mdg_fnd_bus_doc_msg_header WITH UNIQUE KEY id-content.`
- Implement method `IF_MDG_SE_BP_BULK_REPLRQ_IN~INBOUND_PROCESSING` to store the message header of the current single message within the class table attribute.
- Implement all other (currently only one) methods empty or with a `RETURN.`

2.4.2 Implementation for BAdI MDG_SE_BP_BULK_REPLRQ_OUT

If records shall be created and changed from business users on the ADOCM system and sent to the CDO system (so-called client maintenance), it might happen in cases of identified duplicates on ADOCM system, that in some cases, changes from the ADOCM system are not processed on the CDO system due to message sequencing of records. On the sender side, the business partner ID is used to determine the next valid value for changeOrdinalNumberValue. On the sender side however the sequencing context is made up by the receiver business partner UUID and the sent changeOrdinalNumberValue. In case of duplicates on a system this means the changeOrdinalNumberValue is determined by their respective business partner ID, whereas on receiver side the business partner UUID of the best record and the sent changeOrdinalNumberValue is taken to determine the correct sequence which can lead to conflicts.

So, it should be ensured on the sender side that for the changeOrdinalNumberValue always the highest changeOrdinalNumberValue of all identified duplicates of the same best record should be used.

2.4.3 Implementation for BAdI MDG_BS_SUPPLIER_SI

If the message has been sent in the context of a Federation process from a CDO system, the business partner must be written into an MDG, Central Governance Change Request. In all other cases, the business partner shall be activated directly, if it contains no errors. Therefore, the DRF Customizing for the ADO system must be configured as described in the Section “DRF Customizing Settings”.

To achieve the described behavior, implement BAdI MDG_BS_SUPPLIER_SI (Enhancement Spot MDG_BS_SUPPLIER_SI) in your ADOCM system.

The implementing class should:

- Implement method `IF_EX_MDG_BS_SUPPLIER_SI~CHECK_SUPPLIER_DATA_IN` to add an error message to the changing parameter `ct_return_messages` if message has been sent in a Federation context. This ensures that an MDG, Central Governance Change Request is used as persistence. Therefore, the class attributes
  - `gs_admin_message_data` of `cl_mdg_bp_bupa_si_in` and
  - `messageHeaders` of your implementation of BAdI MDG_SE_BP_BULK_REPLRQ_IN can be used.
- Implement all other methods empty or with a `RETURN.`

2.4.4 Enhancement for class CL_MDG_BP_2_STAGING method IF_MDG_BP_2_STA~GET_DEFAULT_CR_TYPE

If for Federation processes certain MDG, Central Governance Change Request types shall be used on the ADO classic mode system, an enhancement implementation of an implicit enhancement option is required at the beginning of method implementation IF_MDG_BP_2_STA~GET_DEFAULT_CR_TYPE, class CL_MDG_BP_2_STAGING. If the incoming message is in the context of a Federation process it can be determined by the sender system and the provided BusinessScope content of the single MessageHeader.

The enhancement implementation should

- use the import parameter `iv_sender_system` and
- use the class attribute `messageHeaders` of your implementation of BAdI MDG_SE_BP_BULK_REPLRQ_IN

to set the return parameter `rv_change_request_type` to the desired MDG, Central Governance Change Request type and then leave the method processing in case the of Federation process context immediately.

2.4.5 Enhancement for class CL_MDG_BP_BUPA_SI_IN method GET_OBJECT_KEY

Since the Federation process uses the SOAP Service BusinessPartnerSUITEBulkReplicateRequest, you must ensure that the correct message order sequencing is also ensured in that case. Therefore, an enhancement implementation of an implicit enhancement option at the beginning of method implementation GET_OBJECT_KEY, class CL_MDG_BP_BUPA_SI_IN, is required.

The enhancement implementation should check the import parameter `is_input_xi` if the BusinessScope is filled in by the sender and indicates an involvement in a Federation process. If this is the case, the return parameter `rv_objkey` should be made up by the BusinessScope-InstanceID and the BusinessPartner-UUID. Leave the method processing in case of the Federation process context immediately.

3 Additional Information

3.1 Further Reading

3.1.1 Information on SAP MDG on SAP S/4HANA

- Exchange knowledge: SAP Community | Q&A | Blog
- Try SAP Master Data Governance on S/4HANA for free: Trial Version
- Try SAP Master Data Governance on S/4HANA on the SAP Cloud Appliance Library: S/4HANA 2022 FPS1
- Learn more: Latest Release | Help Portal | How-to Information | Key Presentations

3.1.2 SAP Roadmap Explorer

- Please see the roadmap for SAP Master Data Governance

3.1.3 Related Information

- Learn more: Floorplan Manager for Web Dynpro ABAP | How to Adapt FPM | FPM Blog | How-to Information | Service Mapping Tool | SAP S/4HANA Cookbook CVI

3.2 SAP Notes

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

Note	Description
3379338	MDG Federation: Application data owners in classic mode - SAP S/4HANA 2023 FPS01

3.3 SAP Source Code Examples

3.3.1 BAdI Implementation for BAdI MDG_SE_BP_BULK_REPLRQ_IN

```
CLASS zcl_bp_bulk_req_in_ado_classic DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES if_badi_interface .
    INTERFACES if_mdg_se_bp_bulk_replrq_in .

    CLASS-DATA messageHeaders TYPE SORTED TABLE OF mdg_fnd_bus_doc_msg_header WITH UNIQUE KEY id-content.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.


CLASS zcl_bp_bulk_req_in_ado_classic IMPLEMENTATION.
  METHOD if_mdg_se_bp_bulk_replrq_in~inbound_processing.
    INSERT in-message_header INTO TABLE messageHeaders.

    CHECK in-message_header-sender_business_system_id = <CDOBusinessSystem>
      AND out-partner-header-object_task = 'U'.

    TRY.
        cl_mdg_id_matching_api_bs=>get_instance(
```

```
      EXPORTING iv_direct_db_insert       = abap_false
                iv_set_lcl_system_by_api  = abap_false
      IMPORTING er_if_mdg_id_matching_api = DATA(id_matching_api)
                ev_lcl_business_system    = DATA(own_system)
    ).
  CATCH cx_root INTO DATA(matching_api_exception).
    RETURN.
ENDTRY.

CHECK id_matching_api IS BOUND.
DATA bp_uuid TYPE mdg_object_id_bs.
TRY.
    cl_gdt_conversion=>guid_outbound(
      EXPORTING im_guid_c = out-partner-header-object_instance-bpartnerguid
      IMPORTING
        ex_value  = bp_uuid
    ).
  CATCH cx_gdt_conversion INTO DATA(conversion_exception).
    RETURN.
ENDTRY.
DATA(search_object_key) = VALUE mdg_s_object_key_bs(
  object_type_code                          = if_mdg_otc_const=>bpartner
  identifier_key-ident_defining_scheme_code = if_mdg_idsc_const=>bpartner_uuid
  identifier_key-business_system_id         = own_system
  identifier_key-id_value                   = bp_uuid
).

TRY.
    id_matching_api->get_matching(
```

```
      EXPORTING is_search_key            = search_object_key
      IMPORTING es_matching_objects_easy = DATA(matching_objects)
    ).
  CATCH cx_root INTO matching_api_exception.
    RETURN.
ENDTRY.

LOOP AT matching_objects-matching_objects REFERENCE INTO DATA(matching_object)
 WHERE object_type_code = if_mdg_otc_const=>bpartner AND business_system_id = own_system.
  READ TABLE matching_object->object_identifier
    WITH KEY ident_defining_scheme_code = if_mdg_idsc_const=>bpartner_uuid
             id_value = bp_uuid
    TRANSPORTING NO FIELDS.
  CHECK sy-subrc = 0.
  DATA(own_records_in_matchgroup) = matching_object.
  EXIT.
ENDLOOP.

LOOP AT matching_objects-matching_objects REFERENCE INTO matching_object
 WHERE object_type_code = if_mdg_otc_const=>bpartner AND business_system_id = <CDOBusinessSystem>.
  READ TABLE matching_object->object_identifier
    WITH KEY ident_defining_scheme_code = if_mdg_idsc_const=>bpartner_uuid
             id_value = in-business_partner-uuid-content
    TRANSPORTING NO FIELDS.
  CHECK sy-subrc = 0.
  DATA(cdo_records_in_matchgroup) = matching_object.
  EXIT.
ENDLOOP.
IF matching_objects-no_matching_objects_found = abap_true
```


```
    OR own_records_in_matchgroup IS BOUND AND lines( own_records_in_matchgroup->object_identifier ) < 2
    OR cdo_records_in_matchgroup IS BOUND AND lines( cdo_records_in_matchgroup->object_identifier ) < 2.
     cl_mdg_ukm=>set_foreign_objects(
       EXPORTING iv_foreign_system_id   = in-message_header-sender_business_system_id
                 iv_foreign_bupa_id     = in-business_partner-internal_id
                 iv_foreign_bupa_uuid   = in-business_partner-uuid-content
                 iv_foreign_supplier_id = in-business_partner-supplier-internal_id
                 iv_foreign_customer_id = in-business_partner-customer-internal_id
                 iv_own_bupa_id         = out-partner-header-object_instance-bpartner
                 iv_own_bupa_uuid       = out-partner-header-object_instance-bpartnerguid
                 iv_own_supplier_id     = out-vendor-header-object_instance-lifnr
                 iv_own_customer_id     = out-customer-header-object_instance-kunnr
                 iv_no_save             = abap_true
       EXCEPTIONS
         error                = 1
         OTHERS               = 2
     ).
     IF sy-subrc <> 0.
       RETURN.
     ENDIF.
   ENDIF.
 ENDMETHOD.

 METHOD if_mdg_se_bp_bulk_replrq_in~outbound_processing.
   RETURN.
 ENDMETHOD.
ENDCLASS.
```

Replace <. . .> with the respective business system.

3.3.2 BAdI Implementation for BAdI MDG_SE_BP_BULK_REPLRQ_OUT

```
CLASS zcl_bp_bulk_req_outado_classic DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_badi_interface .
    INTERFACES if_mdg_se_bp_bulk_replrq_out .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.



CLASS zcl_bp_bulk_req_outado_classic IMPLEMENTATION.
  METHOD if_mdg_se_bp_bulk_replrq_out~outbound_processing.
    CHECK out-business_partner-receiver_uuid-content IS NOT INITIAL
      AND out-business_partner-change_ordinal_number_value IS NOT INITIAL.
    TRY.
        cl_mdg_id_matching_api_bs=>get_instance(
          EXPORTING iv_direct_db_insert       = abap_false
                    iv_set_lcl_system_by_api  = abap_false
          IMPORTING
            er_if_mdg_id_matching_api = DATA(id_matching_api)
        ).
      CATCH cx_root INTO DATA(matching_api_exception).
```

```
    RETURN.
ENDTRY.

CHECK id_matching_api IS BOUND.
DATA(search_object_key) = VALUE mdg_s_object_key_bs(
  object_type_code                          = if_mdg_otc_const=>bpartner
  identifier_key-ident_defining_scheme_code = if_mdg_idsc_const=>bpartner_uuid
  identifier_key-business_system_id         = out-message_header-recipient_business_system_id
  identifier_key-id_value                   = out-business_partner-receiver_uuid-content
).

TRY.
    id_matching_api->get_matching(
      EXPORTING is_search_key            = search_object_key
                iv_target_system         = out-message_header-sender_business_system_id
      IMPORTING es_matching_objects_easy = DATA(matching_objects)
    ).
  CATCH cx_root INTO matching_api_exception.
    RETURN.
ENDTRY.

CHECK matching_objects-matching_objects IS NOT INITIAL
  AND matching_objects-no_matching_objects_found = abap_false.
SELECT COUNT( * ) FROM @matching_objects-matching_objects AS matchingObjects
 WHERE object_type_code = @if_mdg_otc_const=>bpartner
 INTO @DATA(records_in_matchgroup) ##itab_key_in_select.
CHECK records_in_matchgroup > 1.

LOOP AT matching_objects-matching_objects REFERENCE INTO DATA(matching_object)
```


```
      WHERE object_type_code = if_mdg_otc_const=>bpartner.
      DATA(own_duplicate_partner) =
       VALUE #( matching_object->object_identifier[ ident_defining_scheme_code = if_mdg_idsc_const=>bpartner_nr ]-
id_value OPTIONAL ).
      CHECK own_duplicate_partner IS NOT INITIAL
        AND own_duplicate_partner <> out-business_partner-internal_id.
        DATA partner_number TYPE bu_partner.
        partner_number = |{ own_duplicate_partner ALPHA = IN }|.
        TRY.
            cl_bs_soa_inappseq_out=>get_next_message_number(
              EXPORTING iv_obj_type                  = 'BUS1006'
                        iv_sequencing_context        = |BusinessPartnerSUITEReplicateRequestMessage_{
partner_number }|
                        iv_appl_message_id           = out-message_header-id-content
                        iv_recipient_business_system = out-message_header-recipient_business_system_id
              IMPORTING ev_message_number            = data(message_number)
          ).
          CATCH cx_bs_soa_exception INTO DATA(soa_exception).
            CONTINUE.
        ENDTRY.

      CHECK message_number > out-business_partner-change_ordinal_number_value.
      out-business_partner-change_ordinal_number_value = |{ message_number ALPHA = OUT }|.
    ENDLOOP.
  ENDMETHOD.
ENDCLASS.
```

3.3.3 BAdI Implementation for BAdI MDG_BS_SUPPLIER_SI

```
CLASS zcl_bs_supplier_ado_classic DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_badi_interface .
    INTERFACES if_ex_mdg_bs_supplier_si .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.


CLASS zcl_bs_supplier_ado_classic IMPLEMENTATION.
  METHOD if_ex_mdg_bs_supplier_si~check_supplier_data_in.
    CHECK cl_mdg_bp_bupa_si_in=>gs_admin_message_data-single_message_header_id IS NOT INITIAL.
    DATA(currentSingleMessageHeader) =
      VALUE #(
        zcl_bp_bulk_req_in_ado_classic=>messageHeaders[
            id-content = cl_mdg_bp_bupa_si_in=>gs_admin_message_data-single_message_header_id
        ] OPTIONAL
      ).
    CHECK currentSingleMessageHeader IS NOT INITIAL
      AND currentSingleMessageHeader-business_scope-instance_id-content IS NOT INITIAL
      AND currentSingleMessageHeader-business_scope-id-content IS NOT INITIAL
      AND (   currentSingleMessageHeader-business_scope-type_code-content = 'MRG'
```

```
          OR currentSingleMessageHeader-business_scope-type_code-content = 'MRA'
          OR currentSingleMessageHeader-business_scope-type_code-content = 'APP'
          OR currentSingleMessageHeader-business_scope-type_code-content = 'APA'
         ).
    APPEND VALUE #( type = 'E' message = 'Federation process involved; trigger change request' )
        TO ct_return_messages.
  ENDMETHOD.

  METHOD if_ex_mdg_bs_supplier_si~save_supplier_data_in.

  ENDMETHOD.

ENDCLASS.
```

3.3.4 Enhancement for class CL_MDG_BP_2_STAGING method IF_MDG_BP_2_STA~GET_DEFAULT_CR_TYPE

```
IF iv_sender_system = <CDOBusinessSystem>.
  DATA(firstSingleMessageHeader) = VALUE #( zcl_bp_bulk_req_in_ado_classic=>messageHeaders[ 1 ] OPTIONAL ).
  IF firstSingleMessageHeader IS NOT INITIAL
    AND firstSingleMessageHeader-business_scope-instance_id-content IS NOT INITIAL
    AND firstSingleMessageHeader-business_scope-id-content IS NOT INITIAL
    AND (   firstSingleMessageHeader-business_scope-type_code-content = 'MRG'
         OR firstSingleMessageHeader-business_scope-type_code-content = 'MRA'
         OR firstSingleMessageHeader-business_scope-type_code-content = 'APP'
         OR firstSingleMessageHeader-business_scope-type_code-content = 'APA'
        ).
    IF iv_partner_object_task = gc_object_task_insert.
      rv_change_request_type = <SingleObjectChangeRequestType_New>.
    ELSEIF iv_partner_object_task = gc_object_task_update
     OR iv_partner_object_task = gc_object_task_modify.
      rv_change_request_type = <SingleObjectChangeRequestType_Update>.
    ELSE.
      rv_change_request_type = <MultipleObjectChangeRequestType>.
    ENDIF.
    RETURN.
  ENDIF.
ENDIF.
```

Replace all <...> with the respective business system name or rather desired change request types.

3.3.5 Enhancement for class CL_MDG_BP_BUPA_SI_IN method GET_OBJECT_KEY

```
  IF is_input_xi-message_header-business_scope IS NOT INITIAL
   AND ( is_input_xi-message_header-business_scope-type_code-content = 'MRG'
        OR is_input_xi-message_header-business_scope-type_code-content = 'MRA'
        OR is_input_xi-message_header-business_scope-type_code-content = 'APP'
        OR is_input_xi-message_header-business_scope-type_code-content = 'APA'
       ).
    rv_objkey =
|{ is_input_xi-message_header-business_scope-instance_id-content }_{ is_input_xi-business_partner-uuid-content }|.
    RETURN.
  END
```