SAP NetWeaver Master Data Management (MDM) Posting Confirmation via Exchange Infrastructure

Applies to:

SAP NetWeaver Master Data Management 5.5; SAP R/3; SAP ECC; Exchange Infrastructure

Summary

In the Central Master Data Management scenario, there is no information available within SAP NetWeaver MDM about the posting status in backend system(s). There is often the need to check either in MDM or the portal as well as sending notifications to interested parties. This document will describe a scenario of how to implement a ccBPM and programs in a backend system (SAP ECC) to track the confirmation of posting.

Author: Steffen Nesper

Company: SAP (Switzerland) AG

Created on: 20. December 2006

Author Bio

Steffen Nesper is a consultant focusing on process integration with SAP Exchange Infrastructure. He has also been working in the area of Business Connector and Master Data Management for more then 3 years now.

Table of Contents

Functional Description

In the first part of this document, the business need and the process flow between MDM via XI and ECC will be described.

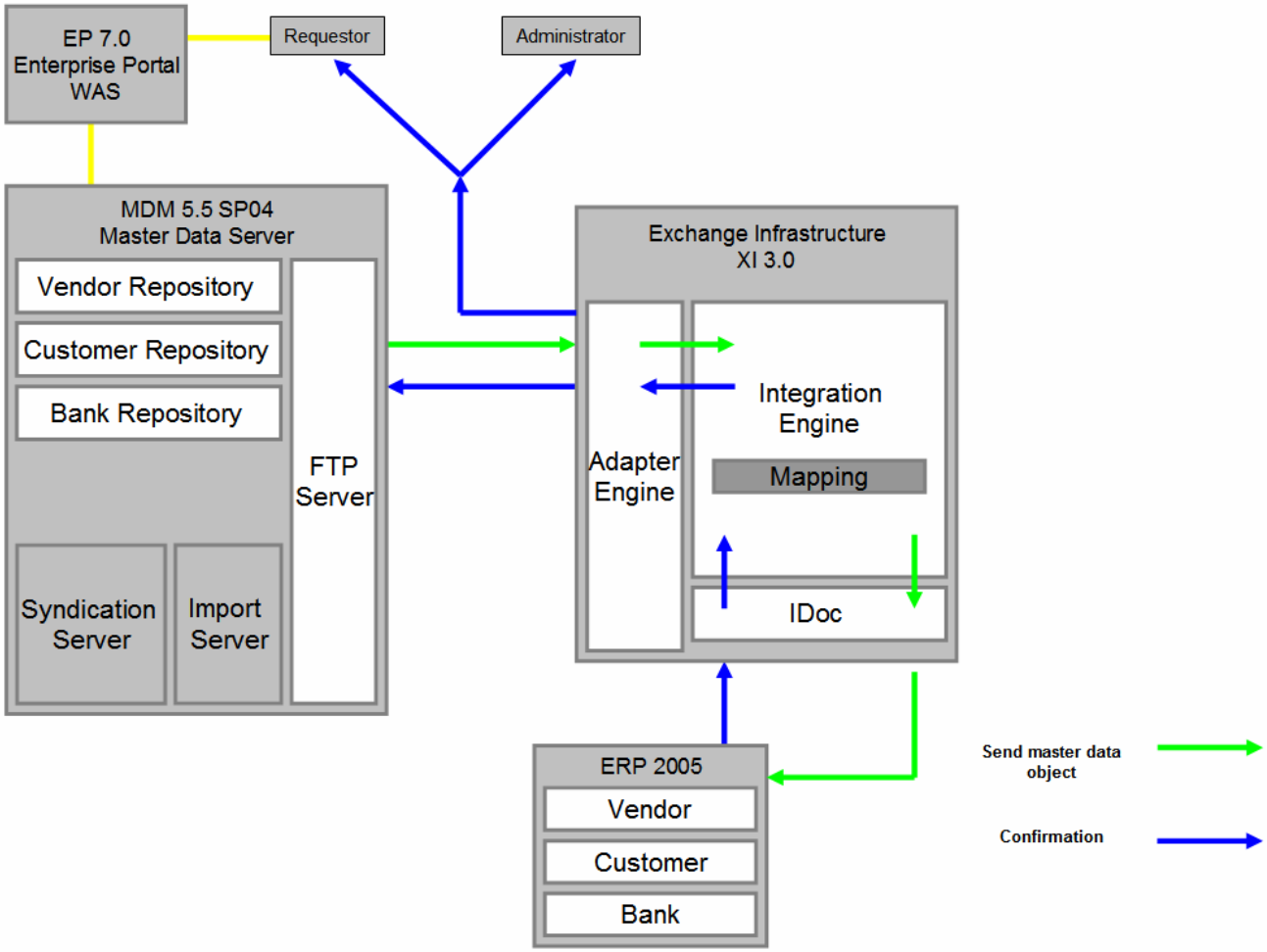
Functional Specification

SAP NetWeaver Master Data Management 5.5 (MDM) distributes master data to SAP ERP Central Component (ECC). As MDM does not have a direct interface into SAP ECC, SAP Exchange Infrastructure (XI) will be used for distribution of the master data objects. This could be vendors, customers or banks or any other master data object. In a central master data management scenario, there is no feedback provided by the backend system if the distribution was successful or not. If this is mandatory in your project, the following process might be considered.

There will be a complex process developed via Business Process Management (BPM) to be able to send master data into SAP and query the delivery status and post it back to MDM in case of success. Also the process needs to send emails to the requestor of the master data change/creation in case of success or send an email to a general administrator email address in case of failure.

In the described business case, there will be a portal view to enter master data requests via SAP Enterprise Portal through a person called “requestor”. This article will focus on the implementation of the data flow between MDM, XI and ECC.

Interface Process Flow Diagram



Transfer Method/Program

MDM delivers the output files in XML format. XI needs to transform it to SAP conform standard. If you do not have additional fields and custom developments, you're able to use the SAP standard IDocs CREMAS and DEBMAS for Vendor and Customer distribution. For international address versions the ADRMAS IDoc will also be used. Also some values of the standard address might not be covered in the CREMAS or DEBMAS IDoc. It is always wise, to use the ADRMAS for address distribution.

For performance and stability reasons we do not want to use a synchronous communication type. It might also not possible in case we need to distribute to n backend systems. Therefore the response of successful posting in SAP needs to be sent separately. There is a standard IDoc Type (ALEAUD) which has all necessary information we need. However, XI and BPM are not able to use this IDoc due to design restrictions. We decided to copy the standard IDoc ALEAUD to a custom IDoc ZALEAUD01 with exactly the same information and some minor modifications to the standard code.

Also to consider the high data volume during initial load / migration, there is the need for a separate load mechanism, so we need to distinguish between initial load and operational mode. For operational mode, we decided to use BPM because we needed the confirmation out of ECC and sending emails to the requestor. This can only be handled via workflows. For Banks, this is not necessary, so only customers and vendors master data processes need to be distinguished and are considered in this article.

Case initial load

MDM will extract the master data objects (vendor, customer, bank) into a folder on the MDM server as XML-files. XI will pick it up via file adapter, map to the correct format and send to ECC system(s). No response from ECC system expected. ECC system has to be monitored via BD87 if everything is posted correctly. To be recognized as initial load, a field 'ECC_system' with value 'INI' has to be present within the message payload.

Case operational mode

MDM will extract the master data objects (vendor, customer) into a folder on the MDM server as XML-files. XI will pick them up via file adapter and send it to the BPM process instance (IP_MDM_Vendor_to_SAP / IP_MDM_Customer_to_SAP).

Within that instance, it will correlate for vendor/customer ID and send the master data object to ECC. The process will wait for a certain time (15 minutes) for confirmation messages from ECC. In ECC, there is a report scheduled (ZRBDSTATE) which will listen on the customer/vendor messages and send a status of success back to MDM BPM.

When the business partner object was posted correctly (IDoc status 53) within SAP ECC, the report will periodically send the status confirmation back to XI where it will be picked up by the BPM. If the BPM received all status reports of ECC (in case of multiple backend systems) it will send a confirmation message back to the MDM server in XML format and also send an email to the requestor (email address provided within the payload from MDM) of successful posting in ECC. In case not all business partners were posted correctly in ECC, the process doesn't receive the required positive acknowledgements and will run in the timeout of 15 minutes (configuration possible). Then it will send an email to the requestor if it was posted in least one system and state within the email in which systems the business partner was posted. Also it will send an email to an administrative user who has to check why posting wasn't possible (data error, system error).

After those messages are sent out, the business partner is distributed and the process will end.

For simplicity, only the process for customer master data distribution will be described in detail but can be adapted to vendor etc. very easily.

Value Mapping

Some values which could be changed over time have been defined in a so called “value mapping table” within the directory of Exchange Infrastructure. The following parameters are customizable:

Source	Target	Description
SAP_MDM	SAP_ECC	Agency used
backendSystemCount	1	Value which tells the BPM in how many ECC systems it will send the Business Partner and also the amount of AUDITS it will receive
customerFailedEmailSubject	SystemEmail: Customer request was NOT posted in ECC	If a customer posting failed in ECC, the subject of the Email can be specified with this parameter
vendorFailedEmailSubject	SystemEmail: Vendor request was NOT posted in ECC	If a vendor posting failed in ECC, the subject of the Email can be specified with this parameter
mdmEmailSender	MDM_admin@yourcompany.com	The email address which will be used as sender and reply to address in the email confirmation to the administrator and the requestor
adminEmailReceiver	admin@yourcompany.com	If the posting of a business partner fails in ECC, it will send a message to the person/team specified with this email address

Processing and operational considerations

As already stated earlier, for initial load BPM must not be used as the high volume would open up to many process instances and the confirmation wouldn’t arrive in time. Then the processes would time out and send wrong information of unsuccessful posting messages to the administrator. Also during initial load, there is probably no email address of the requestor specified and all the email messages would fail on the adapter.

Interface Design

To have the full process up and running, there is of course the need to syndicate the master data object in MDM and send it to XI to start the below mentioned process. Also when the confirmation will be sent back from ECC and XI, this information needs to be placed in the inbound port of MDM and imported via import manager. However, this part won't be described within this document in detail.

Exchange Infrastructure Design

The process for customer distribution is split into two parts: initial load via "normal" XI distribution and operational mode via BPM (Business Process Management). Therefore the message payload has to contain a field called "ECC_System" with content "INI" for initial load or "ALL" for operational mode. Also values for single ECC systems are possible.

Initial load

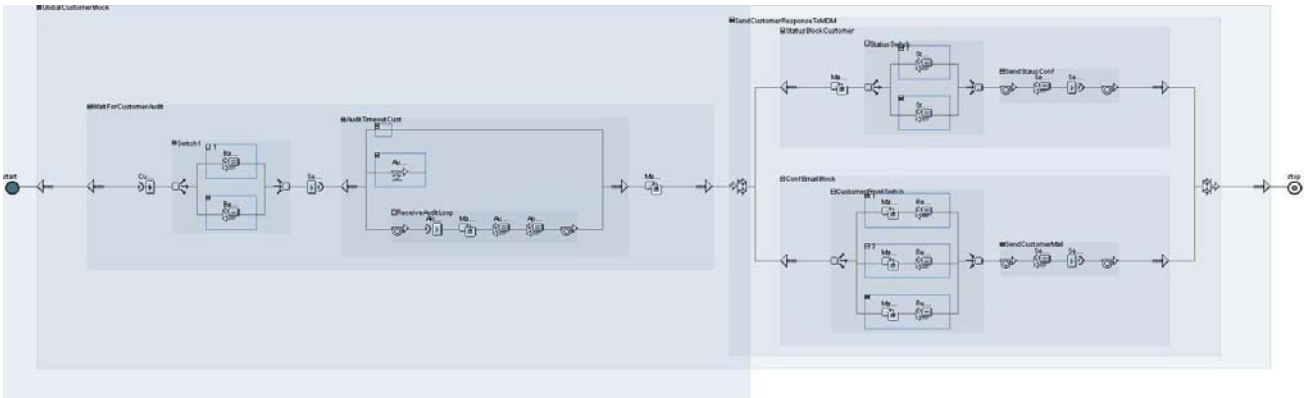
In case of initial load we've got one source message CUSTOMERSMSG containing all data which will be split into two destination IDocs DEBMAS06 and ADRMAS02. DEBMAS contains the customer master data records and ADRMAS contains the address record(s) of the customer.

Operational mode

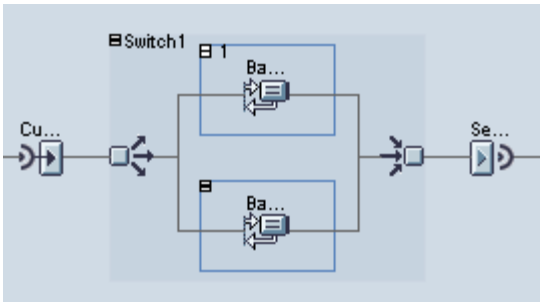
In case of operational mode, BPM will be used. This case is way more complex then the initial load. First of all the customer message will be picked up via the file adapter from the MDM FTP server and send to the Integration process (BPM) IP_MDM_Customer_to_SAP. There it will be correlated and for each customer ID it will start a new process instance which waits for several minutes (default 15) until the confirmation(s) from the backend systems (default 1; changeable via value mapping) will arrive. If the process has received all confirmations from the backend system, it will send an email to the requestor of the customer – the email address has to be provided in the message payload via MDM – and a confirmation message back to MDM with technical status and posting backend system. If the process hasn't received all confirmations from backend system, it will run into a timeout and send partial information in case it received at least on confirmation to the requestor via email and to the admin also via email and to MDM via status message or in case it didn't receive an confirmation at all it will only send an email to the administrator.

An initial mapping is needed to make sure the customer ID has the correct format (10 digits) as MDM might not send with leading zeros. Also as stated earlier, there is a variable called "backendsystems" in the integration process which has the number of ECCs. This variable will be feed with the field of the payload which can be filled with the value provided in the value mapping variable "backendSystemCount".

Process overview



Customer message arrives in BMP:



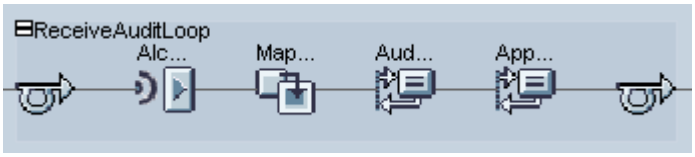
When the message arrives, it will check if it goes to “ALL” backend systems or if it will only go to one dedicated backend system. In case of all, the variable “backendsystems” will be filled via value mapping parameter “backendSystemCount”; it has to be adapted to your system landscape. The information if it is going to one or all backend systems has to be provided via message payload.

In case of only one dedicated backend system, it will of course always send it to one backend system only and nothing has to be customized.

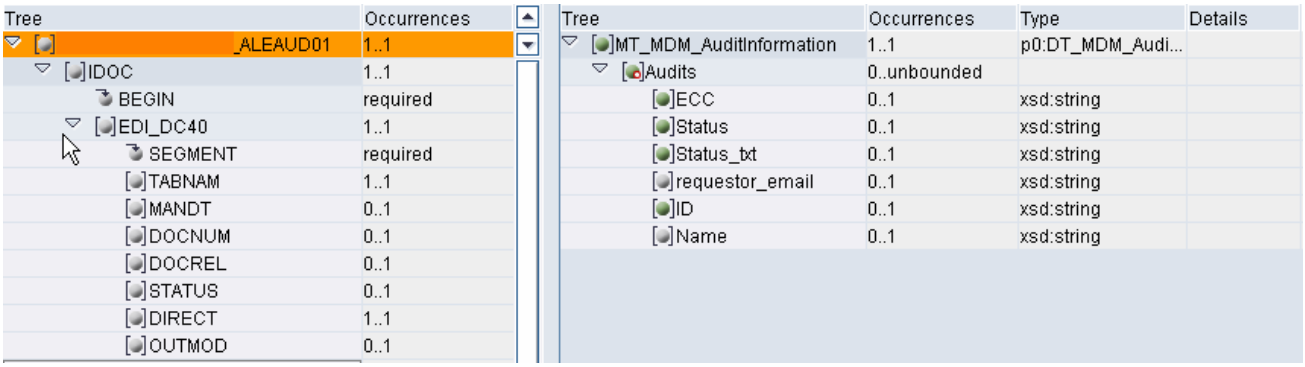
After that first switch and backend system determination the Customer and Address data is sent to ECC.

After the data was sent to ECC, it waits in a loop for response messages from the backend systems. In the example, we are waiting 15 minutes until all audits have arrived. If it does not arrive in time or there was an error, it will end the loop and proceed.

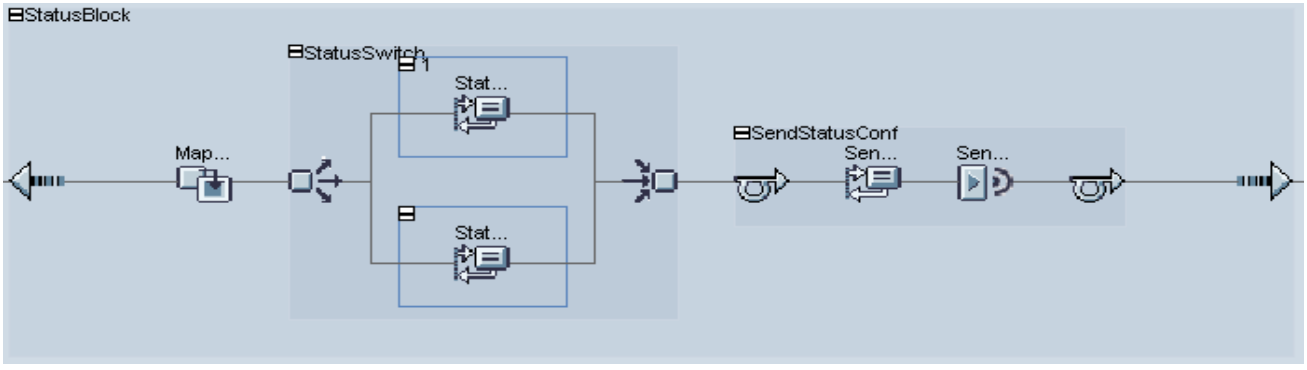
Audit/Confirmation receive from ECC;
Correlation on OBJTYPE=KNA1 and OBJKEY=CustomerID



Mapping of AuditIDoc to internal Audit format (only some key fields necessary; status, ID, backend system)

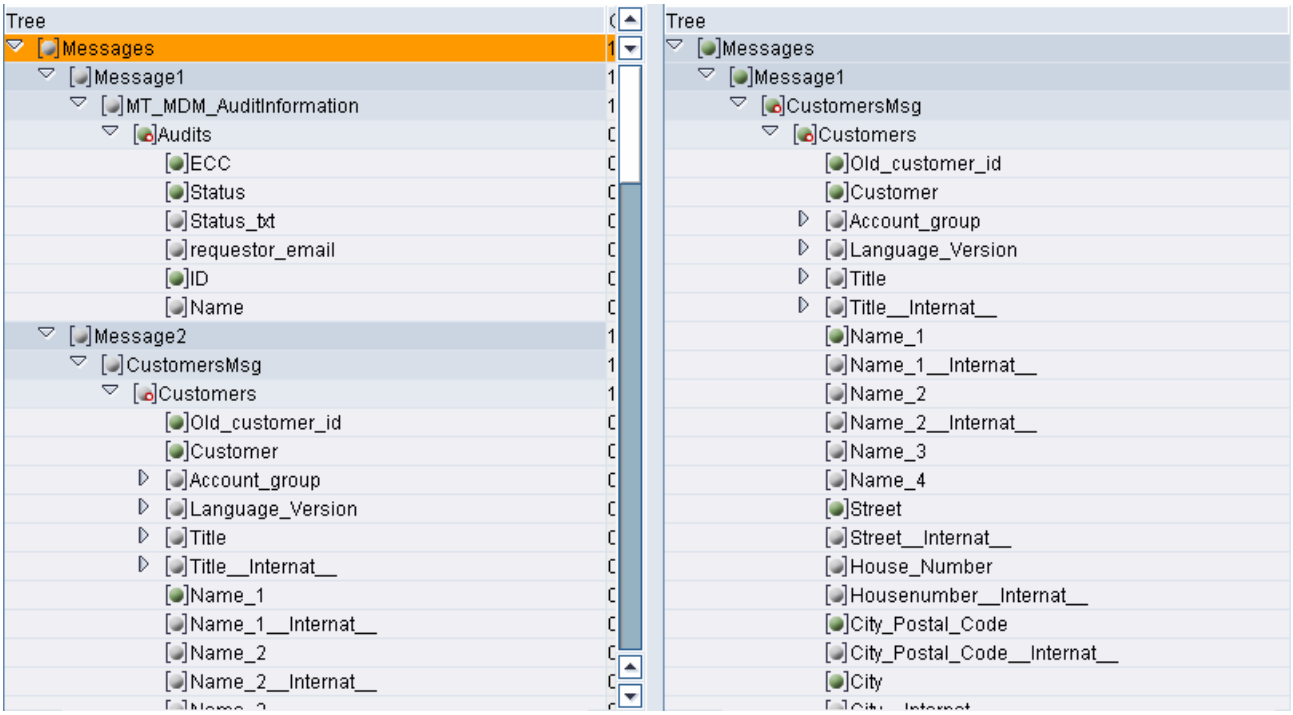


Collecting and mapping the Status information from the audit and send it back to MDM:

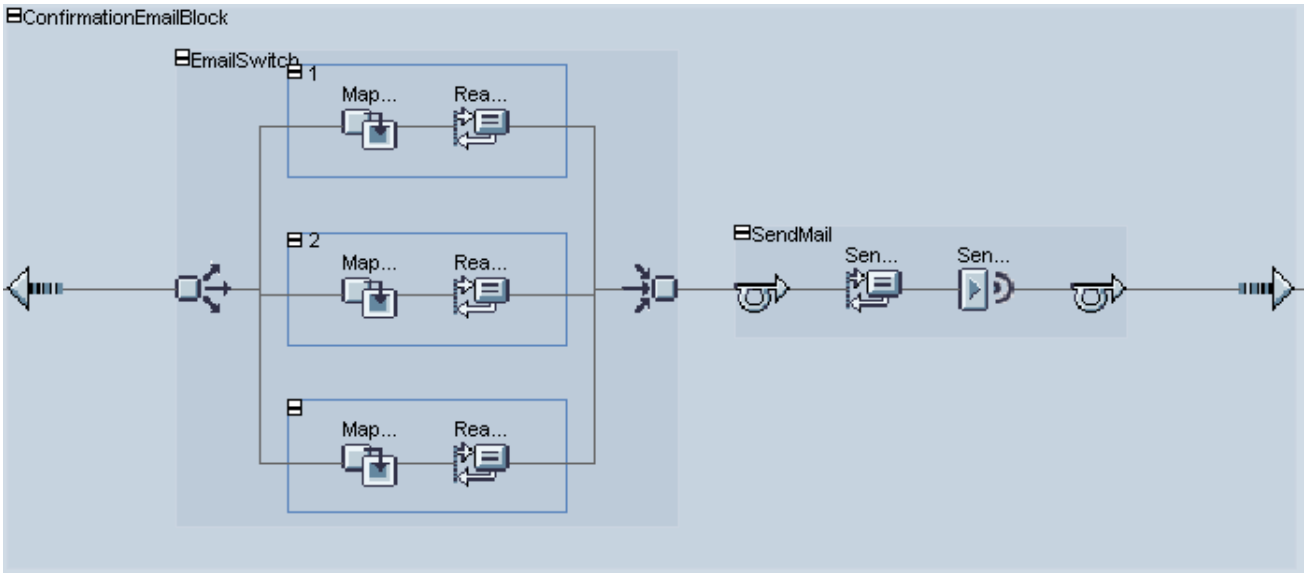


The switch will determine again if all audits are arrived and how many audits have to be mapped together to one message. We only want to send one email or confirmation in case of multiple backend systems and provide a list with all status information.

Here we've got a multi mapping of the source customer file which started the process and the audit information from backend system to the new confirmation message which will be sent to the MDM. For confirmation we used the same XSD as for syndication in the first place (MDM and XI). We enrich the data with the status of the backend system and are able to have an exact matching during import against existing data in the repository.



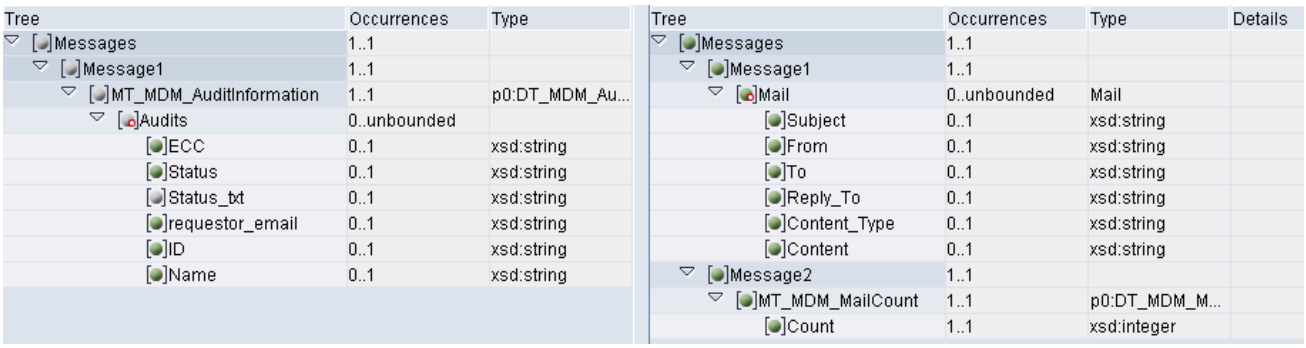
Collecting and Mapping the audit information and send it to the requestor or admin via email:



Dependent on the audit information a different mapping has to be called

Case auditcount = backendsystems:

It received the same number of audit as backend systems maintained, all audits received; Send only one mail to the requestor with success message.

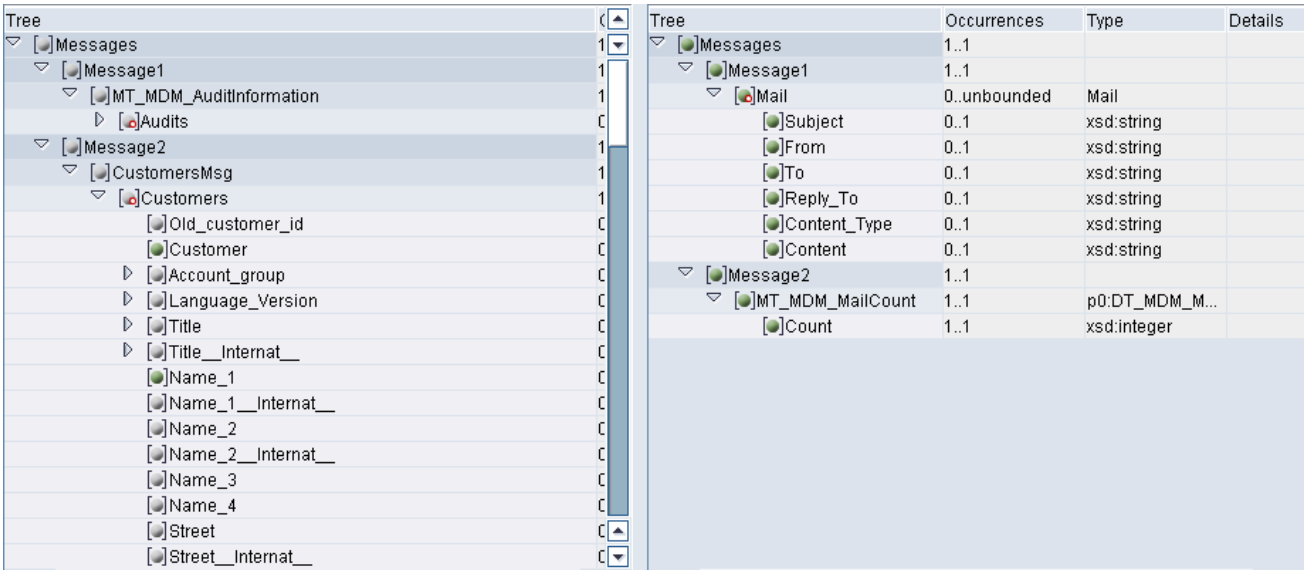


Tree	Occurrences	Type
Messages	1..1	
Message1	1..1	
MT_MDM_AuditInformation	1..1	p0:DT_MDM_Au...
Audits	0..unbounded	
ECC	0..1	xsd:string
Status	0..1	xsd:string
Status_txt	0..1	xsd:string
requestor_email	0..1	xsd:string
ID	0..1	xsd:string
Name	0..1	xsd:string

Tree	Occurrences	Type	Details
Messages	1..1		
Message1	1..1		
Mail	0..unbounded	Mail	
Subject	0..1	xsd:string	
From	0..1	xsd:string	
To	0..1	xsd:string	
Reply_To	0..1	xsd:string	
Content_Type	0..1	xsd:string	
Content	0..1	xsd:string	
Message2	1..1		
MT_MDM_MailCount	1..1	p0:DT_MDM_M...	
Count	1..1	xsd:integer	

Case auditcount = 0:

It has not received any audit message from backend system; Send only one mail to the administrator with failure message



Tree
Messages
Message1
MT_MDM_AuditInformation
Audits
Message2
CustomersMsg
Customers
Old_customer_id
Customer
Account_group
Language_Version
Title
Title__Internat__
Name_1
Name_1__Internat__
Name_2
Name_2__Internat__
Name_3
Name_4
Street
Street__Internat__

Tree	Occurrences	Type	Details
Messages	1..1		
Message1	1..1		
Mail	0..unbounded	Mail	
Subject	0..1	xsd:string	
From	0..1	xsd:string	
To	0..1	xsd:string	
Reply_To	0..1	xsd:string	
Content_Type	0..1	xsd:string	
Content	0..1	xsd:string	
Message2	1..1		
MT_MDM_MailCount	1..1	p0:DT_MDM_M...	
Count	1..1	xsd:integer	

Default case:

It received only a friction of audit messages from backend systems (only if you have more then one backend system for distribution); It will send an mail to the requestor with success message and one mail to the administrator with failure message.

Tree	Occurrences	Type
Messages	1..1	
Message1	1..1	
MT_MDM_AuditInformation	1..1	p0:DT_MDM_Au...
Audits	0..unbounded	
ECC	0..1	xsd:string
Status	0..1	xsd:string
Status_txt	0..1	xsd:string
requestor_email	0..1	xsd:string
ID	0..1	xsd:string
Name	0..1	xsd:string

Tree	Occurrences	Type	Details
Messages	1..1		
Message1	1..1		
Mail	0..unbounded	Mail	
Subject	0..1	xsd:string	
From	0..1	xsd:string	
To	0..1	xsd:string	
Reply_To	0..1	xsd:string	
Content_Type	0..1	xsd:string	
Content	0..1	xsd:string	
Mail	0..unbounded	Mail	
Subject	0..1	xsd:string	
From	0..1	xsd:string	
To	0..1	xsd:string	
Reply_To	0..1	xsd:string	
Content_Type	0..1	xsd:string	
Content	0..1	xsd:string	
Message2	1..1		
MT_MDM_MailCount	1..1	p0:DT_MDM_M...	
Count	1..1	xsd:integer	

Finally, send the email to the requestor and/or to the administrator.

SAP ECC Design

Confirmation Message / Custom IDoc type

Due to design limitations, it is not possible to evaluate the content of the standard ALEAUD IDoc within SAP Exchange Infrastructure. Therefore we needed to tweak XI and use any other type of IDoc message for the confirmation message. However, the content of the ALEAUD is exactly what we need so we decided to copy the standard IDoc ALEAUD01 to a custom IDoc type ZALEAUD with exactly the same structure.

Status Report

For sending ALEAUDs a standard report exists in each ECC already (RBDSTATE), this report will send an ALEAUD to the outbound port of ECC. Within a variant to that report, it has to be specified for what kind of inbound IDocs and the business systems it should react. As that standard report will only send out the standard ALEAUD IDoc types, we also need to copy that report to ZRBDSTATE and change to using IDocs of type ZALEAUD instead. Changes to the standard are marked in yellow.

```
<<<
REPORT ZRBDSTATE MESSAGE-ID b1.

INCLUDE mbdconst.
INCLUDE bdcstaud.

CONSTANTS:
  c_max_idocs TYPE i VALUE '500',
  c_mestyp_aleaud1 LIKE edidc-mestyp VALUE 'ZALEAUD '.


TABLES: edidc, sscrfields, bdaudstate.

DATA: gs_layout TYPE slis_layout_alv,
```



```
    gt_fieldcat TYPE slis_t_fieldcat_alv,
    gs_excluding TYPE slis_t_extab,
    gs_excl_head TYPE slis_extab,
    g_status_set   TYPE slis_formname VALUE 'PF_STATUS_SET',
    g_user_command TYPE slis_formname VALUE 'USER_COMMAND',
    header TYPE lvc_title.

SELECT-OPTIONS:
            s_sndsys FOR bdaudstate-rcv_system.
SELECTION-SCREEN SKIP 1.
SELECT-OPTIONS:
            s_mestyp FOR bdaudstate-mess_type,
            s_mescod FOR bdaudstate-mess_code,
            s_mesfct FOR bdaudstate-mess_funct.
SELECTION-SCREEN SKIP 1.
SELECT-OPTIONS: s_upddat FOR edidc-upddat NO-EXTENSION .

DATA:
    left_date LIKE edidc-upddat,
    left_time LIKE edidc-updtim,
    right_date LIKE edidc-upddat,
    right_time LIKE edidc-updtim,
    t_idoc_control TYPE audit_idoc_control_tab,
    t_idoc_control_all TYPE audit_idoc_control_tab,
    s_idoc_control TYPE audit_idoc_control_t ,
    nothing_to_do,
    resulting_idocs LIKE bdidocs OCCURS 0 WITH HEADER LINE.

DATA: paket TYPE i,
    t_idoc_control_all_max TYPE i,
    t_idoc_control_all_anz TYPE i.


START-OF-SELECTION.

 IF NOT sy-batch IS INITIAL AND s_upddat[] IS INITIAL.
   PERFORM time_interval_get
     CHANGING
       left_date
       left_time
       right_date
```



```
      right_time.

  MESSAGE i147 WITH left_date left_time right_date right_time.
*  Selektionszeitraum für die AUDIT Idocs: &1 &2 -> &3 &4 .

*  comparisons are only based on numbers (date and time). This is
*  portable to AS/400.
  SELECT sndprn mestyp mescod mesfct credat cretim docnum status
         FROM edidc INTO TABLE t_idoc_control_all
   WHERE ( upddat = left_date AND updtim >= left_time  "#EC PORTABLE
   OR upddat > left_date )                             "#EC PORTABLE
   AND ( upddat < right_date                           "#EC PORTABLE
   OR upddat = right_date AND updtim <= right_time )   "#EC PORTABLE
   AND sndprt = c_prt_logical_system
   AND sndprn IN s_sndsys
   AND mestyp IN s_mestyp
   AND mescod IN s_mescod
   AND mesfct IN s_mesfct
   AND status <> c_status_in_archive_reload
   AND status <> c_status_in_archived
   AND status <> c_status_in_orig_of_edited
  ORDER BY sndprn mestyp mesfct mescod credat cretim.

  IF sy-subrc <> 0.
   nothing_to_do = 'X'.
  ELSE.

   DESCRIBE TABLE t_idoc_control_all LINES t_idoc_control_all_max.

   LOOP AT t_idoc_control_all INTO s_idoc_control.

     APPEND s_idoc_control TO t_idoc_control.
     t_idoc_control_all_anz = t_idoc_control_all_anz + 1.
     paket = paket + 1.
     IF paket = c_max_idocs OR
       t_idoc_control_all_anz = t_idoc_control_all_max.

       PERFORM idocs_create
         TABLES
           resulting_idocs
         USING
```



```
          t_idoc_control.

        CLEAR paket.
        REFRESH t_idoc_control.
      ENDIF.
    ENDLOOP.

  ENDIF.
ELSE.

  MESSAGE s147 WITH
  s_upddat-low '00:00:00' s_upddat-high '24:00:00' .
*   Selektionszeitraum für die AUDIT Idocs: &1 &2 -> &3 &4 .

  SELECT sndprn mestyp mescod mesfct credat cretim docnum status
    FROM edidc INTO TABLE t_idoc_control_all
    WHERE upddat IN s_upddat
      AND sndprt = c_prt_logical_system
      AND sndprn IN s_sndsys
      AND mestyp IN s_mestyp
      AND mescod IN s_mescod
      AND mesfct IN s_mesfct
      AND status <> c_status_in_archive_reload
      AND status <> c_status_in_archived
      AND status <> c_status_in_orig_of_edited
    ORDER BY sndprn mestyp mesfct mescod credat cretim.

  IF sy-subrc <> 0.
    nothing_to_do = 'X'.
  ELSE.

    DESCRIBE TABLE t_idoc_control_all LINES t_idoc_control_all_max.

    LOOP AT t_idoc_control_all INTO s_idoc_control.

      APPEND s_idoc_control TO t_idoc_control.
      t_idoc_control_all_anz = t_idoc_control_all_anz + 1.
      paket = paket + 1.
      IF paket = c_max_idocs OR
        t_idoc_control_all_anz = t_idoc_control_all_max.
```



```
        PERFORM idocs_create
          TABLES
            resulting_idocs
          USING
            t_idoc_control.

        CLEAR paket.
        REFRESH t_idoc_control.
       ENDIF.
     ENDLOOP.


   ENDIF.
  ENDIF.

  IF NOT nothing_to_do IS INITIAL.
    MESSAGE i139.
*    Es wurden keine auditrelevanten Daten selektiert.
  ELSEIF NOT resulting_idocs[] IS INITIAL.
    COMMIT WORK.
    CALL FUNCTION 'DEQUEUE_ALL'.
    PERFORM output_list.
  ELSE.
    MESSAGE i158.
*   Für die selektierten Daten gibt es keine Interessenten
  ENDIF.

AT SELECTION-SCREEN ON s_upddat.
  IF sy-batch IS INITIAL AND
     ( sscrfields-ucomm = 'ONLI' OR sscrfields-ucomm = 'PRIN' ).
    IF s_upddat[] IS INITIAL.
      MESSAGE e142.
*    Es muß ein Änderungszeitraum angegeben werden.
    ENDIF.
  ENDIF.

*&---------------------------------------------------------------------*
*&      Form  TIME_INTERVAL_GET
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
```



```
*      <--P_LEFT_DATE  text                                       *
*      <--P_LEFT_TIME  text                                      *
*      <--P_RIGHT_DATE  text                                      *
*      <--P_RIGHT_TIME  text                                      *
*---------------------------------------------------------------------*
FORM time_interval_get
  CHANGING
    left_date LIKE edidc-upddat
    left_time LIKE edidc-updtim
    right_date LIKE edidc-upddat
    right_time LIKE edidc-updtim.

  DATA:
    job_name LIKE tbtcjob-jobname,
    job_count LIKE tbtcjob-jobcount,
    job_head LIKE tbtcjob,
    cmp_time LIKE edidc-updtim.

  CALL FUNCTION 'GET_JOB_RUNTIME_INFO'
       IMPORTING
*        eventid             =
*        eventparm           =
*        external_program_active =
          jobcount            = job_count
          jobname             = job_name.
*        stepcount           =
*    exceptions
*        no_runtime_info     = 1
*        others              = 2.

  CALL FUNCTION 'BP_JOB_READ'
    EXPORTING
      job_read_jobcount = job_count
      job_read_jobname  = job_name
      job_read_opcode   = 19
    IMPORTING
      job_read_jobhead  = job_head.
*    tables
*        JOB_READ_STEPLIST     =
*    exceptions
*        invalid_opcode      = 1
```



```
*       job_doesnt_exist     = 2
*       job_doesnt_have_steps = 3
*       others               = 4.

*    sdltime is the time when the job was scheduled, this is either the
*    time, when the job was put to the batch list or the time when the
*    predeccessor started

*    substracting 5 minutes from the schedule time is the left border
*    of the time interval
  left_time = job_head-sdltime - 300.  " 300 seconds = 5 minutes
  cmp_time = -300.                     "23:55
  IF left_time < cmp_time.
    left_date = job_head-sdldate.
  ELSE.
*      date has to changed
    left_date = job_head-sdldate - 1.
  ENDIF.
*    strttime is the time when the job was started
*    substracting 4 minutes from that time is the right border
*    of the time interval, (one minute overlap)
  right_time = job_head-strttime - 240." 240 seconds = 4 minutes
  cmp_time = -240.                     "23:56
  IF right_time < cmp_time.
    right_date = job_head-strtdate.
  ELSE.
*      date has to be changed
    right_date = job_head-strtdate - 1.
  ENDIF.

ENDFORM.                             " TIME_INTERVAL_GET
*&---------------------------------------------------------------------*
*&      Form  IDOCS_CREATE
*&---------------------------------------------------------------------*
*      text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM idocs_create
  TABLES
```



```
    resulting_idocs STRUCTURE bdidocs
  USING
    idoc_controls TYPE audit_idoc_control_tab.

  CONSTANTS:
    c_filter_mestyp LIKE tbd10-objtype VALUE 'ZMESTYP01'.

  DATA: idoc_control TYPE audit_idoc_control_t,
        receiver_input LIKE bdi_logsys OCCURS 0 WITH HEADER LINE,
        receiver_output LIKE bdi_logsys OCCURS 0 WITH HEADER LINE,
        filter_objects LIKE bdi_fltval OCCURS 0 WITH HEADER LINE,
        to_send TYPE c,
        idoc_info TYPE audit_idoc,
        control_package TYPE audit_idoc_tab.

  DATA lt_resulting_idocs TYPE TABLE OF bdidocs.
**************************

  LOOP AT idoc_controls INTO idoc_control.
    AT NEW mestyp.
      REFRESH receiver_input.
      receiver_input-logsys = idoc_control-sndprn.
      APPEND receiver_input.

      REFRESH filter_objects.
*     append entry for filter object MESTYP
      filter_objects-objtype = c_filter_mestyp.
      filter_objects-objvalue = idoc_control-mestyp.
      APPEND filter_objects.

      CALL FUNCTION 'ALE_MESTYPE_GET_RECEIVER'
        EXPORTING
          message_type        = c_mestyp_aleaud01
        TABLES
          receiver_input      = receiver_input
          receivers           = receiver_output
          filterobject_values = filter_objects.
*     exceptions
*         mestype_not_found       = 1
*         error_in_filterobjects  = 2
*         error_in_ale_customizing = 3
```



```
*         others                   = 4.

    IF NOT receiver_output[] IS INITIAL.
      to_send = 'X'.
    ELSE.
      CLEAR to_send.
    ENDIF.
  ENDAT.

  IF NOT to_send IS INITIAL.
    MOVE-CORRESPONDING idoc_control TO idoc_info.
    APPEND idoc_info TO control_package.
  ENDIF.

  AT END OF sndprn.
    IF NOT control_package[] IS INITIAL.
      CALL FUNCTION 'ZAUDIT_IDOC_CREATE'
        EXPORTING
          rcv_system        = idoc_control-sndprn
*>>>>>>>>>>>>>>>>>>>>>>>>  START OF DELETION >>>>>>>>>>>>>>>>>>>>>>>*
***         IMPORTING
***           idoc_number       = resulting_idocs-docnum
*<<<<<<<<<<<<<<<<<<<<<<<<< END OF DELETION <<<<<<<<<<<<<<<<<<<<<<<<*
*>>>>>>>>>>>>>>>>>>>>>>>  START OF INSERTION >>>>>>>>>>>>>>>>>>>>>>>*
        TABLES
          et_resulting_idocs = lt_resulting_idocs
*<<<<<<<<<<<<<<<<<<<<<<<<  END OF INSERTION <<<<<<<<<<<<<<<<<<<<<<<<*
        CHANGING
          idoc_info_records = control_package.


      REFRESH control_package.
      APPEND LINES OF lt_resulting_idocs TO resulting_idocs[].
    ENDIF.
  ENDAT.
 ENDLOOP.

ENDFORM.                            " IDOCS_CREATE
*&---------------------------------------------------------------------*
*&      Form  output_list
*&---------------------------------------------------------------------*
```



```
*       text
*----------------------------------------------------------------------*
*      -->P_RESULTING_IDOCS  text
*----------------------------------------------------------------------*
FORM output_list.

  header = text-001.
  PERFORM fieldcat_init USING gt_fieldcat[].
  PERFORM layout_init USING gs_layout.
  PERFORM icon_excluding USING gs_excluding.

  CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
    EXPORTING
      i_callback_program       = sy-repid
      i_callback_pf_status_set = g_status_set
      i_callback_user_command  = g_user_command
      i_grid_title             = header
      is_layout                = gs_layout
      it_fieldcat              = gt_fieldcat[]
      it_excluding             = gs_excluding[]
    TABLES
      t_outtab                 = resulting_idocs
    EXCEPTIONS
      program_error            = 1
      OTHERS                   = 2.
  IF sy-subrc <> 0.
  ENDIF.


ENDFORM.                    " output_list
*&----------------------------------------------------------------------*
*&      Form  fieldcat_init
*&----------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->P_GT_FIELDCAT[]  text
*----------------------------------------------------------------------*
FORM fieldcat_init  USING  rt_fieldcat
                           TYPE slis_t_fieldcat_alv.

  DATA: ls_fieldcat TYPE slis_fieldcat_alv.
```



```
  CLEAR ls_fieldcat.
  ls_fieldcat-fieldname     = 'DOCNUM'.
  ls_fieldcat-seltext_l     = text-200.
  ls_fieldcat-outputlen     = '30'.
  APPEND ls_fieldcat TO  rt_fieldcat.


ENDFORM.                    " fieldcat_init
*&---------------------------------------------------------------------*
*&      Form  layout_init
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->P_GS_LAYOUT  text
*----------------------------------------------------------------------*
FORM layout_init USING rs_layout TYPE slis_layout_alv.
*doubleclick
  rs_layout-f2code            = 'IDOC'.
  rs_layout-colwidth_optimize           = 'X'.
ENDFORM.                    " LAYOUT_INIT
*&---------------------------------------------------------------------*
*&      Form  icon_excluding
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->P_GS_EXCLUDING  text
*----------------------------------------------------------------------*
FORM icon_excluding  USING  p_gs_excluding TYPE slis_t_extab.

  REFRESH  p_gs_excluding[].
  gs_excl_head-fcode = '&VEXCEL'.                "Excel
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&AQW'.                   "word
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&GRAPH'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&XXL'.
```



```
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&CRBATCH'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&CRTEMPL'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&XINT'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&URL'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&CRDESIG'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&VLOTUS'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&VCRYSTAL'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&OL0'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&XPA'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&OMP'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.

  gs_excl_head-fcode = '&ILT'.
  APPEND gs_excl_head-fcode TO p_gs_excluding.



ENDFORM.                    " ICON_EXCLUDING

*&---------------------------------------------------------------------*
*&      Form  PF_STATUS_SET
```



```
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM pf_status_set USING rt_extab TYPE slis_t_extab.
  DATA: l_status LIKE sy-pfkey VALUE 'STANDARD'.
* EXCLUDING FCODES GIVEN BY ABAP LISTVIEWER *
  SET PF-STATUS l_status EXCLUDING rt_extab.
ENDFORM.                    " PF_STATUS_SET
*&---------------------------------------------------------------------*
*&      Form  USER_COMMAND
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM user_command USING rf_ucomm    LIKE sy-ucomm
                        rs_selfield TYPE slis_selfield.

  CASE rf_ucomm.
    WHEN 'IDOC'.
      READ TABLE resulting_idocs INDEX rs_selfield-tabindex.
      IF sy-subrc = 0.
        SUBMIT idoc_tree_control WITH docnum = resulting_idocs-docnum
                          AND RETURN.
      ELSE.
        MESSAGE s010.
*   Bitte Cursor richtig positionieren
      ENDIF.
      CLEAR rf_ucomm.
    WHEN OTHERS.
  ENDCASE.
ENDFORM. " USER_COMMAND
>>>
```

Also the standard report will collect all reports and send them together within one IDoc. As there is possibility of customer and vendor messages within one run, we decided to split the IDoc and send one separate IDoc for each report. Therefore the standard function had to be copied and modified as follows:


```
<<<
FUNCTION ZAUDIT_IDOC_CREATE.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(RCV_SYSTEM) LIKE  EDIDC-RCVPRN
*"  TABLES
*"      ET_RESULTING_IDOCS STRUCTURE  BDIDOCS
*"  CHANGING
*"     REFERENCE(IDOC_INFO_RECORDS) TYPE  AUDIT_IDOC_TAB
*"----------------------------------------------------------------------


* Copy of FM AUDIT_IDOC_CREATE
* Responsible: Steffen Nesper, SAP AG
* Creation date: 14.11.2006

  DATA: idoc_control LIKE bdicontrol,
        wa_idoc_info TYPE audit_idoc,
        t_idoc_data LIKE edidd OCCURS 0 WITH HEADER LINE,
          idoc_seg LIKE e1state,
          idoc_obj_seg LIKE e1prtob,
          idoc_hdr_seg LIKE e1adhdr,
  t_comm_control LIKE edidc OCCURS 0 WITH HEADER LINE,
  t_status LIKE edids OCCURS 0 WITH HEADER LINE,
  message_id_len TYPE i,
  resulting_idocs LIKE bdidocs  OCCURS 0 WITH HEADER LINE,
  receivers LIKE bdi_logsys OCCURS 0 WITH HEADER LINE.


* Multiple IDoc creation now, therefore the return parameter was
* turned from a single variable into a table
  CLEAR et_resulting_idocs[].


  CALL FUNCTION 'READ_LINKED_OBJECTS_INBOUND'
    CHANGING
      t_idoc_info = idoc_info_records.

  PERFORM read_status_messages
    TABLES
```



```
    t_status
   USING
    idoc_info_records.

*  IDOC_CONTROL-MESTYP = C_MESTYP_ALEAUD.
*  IDOC_CONTROL-IDOCTP = C_IDOCTP_ALEAUD01.

  idoc_control-mestyp = 'ZALEAUD '.
  idoc_control-idoctp = 'ZALEAUD 01'.

  LOOP AT idoc_info_records INTO wa_idoc_info.
*    IF wa_idoc_info-mestyp <> idoc_hdr_seg-mestyp_lng
*    OR wa_idoc_info-mescod <> idoc_hdr_seg-mescod
*    OR wa_idoc_info-mesfct <> idoc_hdr_seg-mesfct.
      idoc_hdr_seg-mestyp_lng = wa_idoc_info-mestyp.
*     try to fill old message type
      CALL FUNCTION 'IDOC_GET_SHORT_MESTYP'
        EXPORTING
          pi_mestyp             = wa_idoc_info-mestyp
        IMPORTING
          pe_mestyp             = idoc_hdr_seg-mestyp
        EXCEPTIONS
          no_conversion_possible = 1
          OTHERS                 = 2.
      IF sy-subrc <> 0.
        CLEAR idoc_hdr_seg-mestyp.
      ENDIF.
      t_idoc_data-segnam = c_segtyp_e1adhdr.
      t_idoc_data-sdata = idoc_hdr_seg.
      APPEND t_idoc_data.
*    ENDIF.
    READ TABLE t_status WITH KEY docnum = wa_idoc_info-docnum
      BINARY SEARCH.
    IF sy-subrc = 0.
      MOVE-CORRESPONDING t_status TO idoc_seg.
      idoc_seg-stapa1_lng = t_status-stapa1.
      idoc_seg-stapa2_lng = t_status-stapa2.
      idoc_seg-stapa3_lng = t_status-stapa3.
      idoc_seg-stapa4_lng = t_status-stapa4.
      IF t_status-stacod IS INITIAL.
*       try to fill 3.0 message
```



```
      message_id_len = STRLEN( t_status-stamid ).
      IF message_id_len <= 2.
        CONCATENATE t_status-stamqu
                    t_status-stamid
                    t_status-stamno
          INTO idoc_seg-stacod.
      ELSE.
        CLEAR: idoc_seg-stapa1, idoc_seg-stapa2, idoc_seg-stapa3,
        idoc_seg-stapa4.
      ENDIF.
    ENDIF.
  ELSE.
    CLEAR: idoc_seg-stacod, idoc_seg-statxt, idoc_seg-stapa1,
      idoc_seg-stapa2, idoc_seg-stapa3, idoc_seg-stapa4,
      idoc_seg-stamqu, idoc_seg-stamid, idoc_seg-stamno,
      idoc_seg-stapa1_lng, idoc_seg-stapa2_lng,
      idoc_seg-stapa3_lng, idoc_seg-stapa4_lng.
  ENDIF.
  idoc_seg-docnum = wa_idoc_info-prt_docnum.
  idoc_seg-status = wa_idoc_info-status.

  t_idoc_data-segnam = c_segtyp_e1state.
  t_idoc_data-sdata = idoc_seg.
  APPEND t_idoc_data.
  MOVE-CORRESPONDING wa_idoc_info TO idoc_obj_seg.
  t_idoc_data-segnam = c_segtyp_e1prtob.
  t_idoc_data-sdata = idoc_obj_seg.
  APPEND t_idoc_data.

* One AUDIT IDoc for every found IDoc
* Therefore: IDoc creation within the loop
  READ TABLE receivers TRANSPORTING NO FIELDS WITH KEY logsys = rcv_system.
  IF NOT sy-subrc EQ 0.
    receivers-logsys = rcv_system.
    APPEND receivers.
  ENDIF.

  CALL FUNCTION 'ALE_IDOCS_CREATE'
    EXPORTING
      idoc_control  = idoc_control
    TABLES
```



```
      idoc_data     = t_idoc_data
      receivers     = receivers
      created_idocs = resulting_idocs.

   READ TABLE resulting_idocs INDEX 1.
   IF sy-subrc = 0.
     APPEND LINES OF resulting_idocs TO et_resulting_idocs.
     CLEAR t_idoc_data.
     CLEAR t_idoc_data[].
   ENDIF.

 ENDLOOP.

*  IF sy-subrc = 0.
*    receivers-logsys = rcv_system.
*    APPEND receivers.
*    CALL FUNCTION 'ALE_IDOCS_CREATE'
*      EXPORTING
*        idoc_control  = idoc_control
*      TABLES
*        idoc_data     = t_idoc_data
*        receivers     = receivers
*        created_idocs = resulting_idocs.
**          APPLICATION_OBJECTS         =
**      exceptions
**          idoc_input_was_inconsistent = 1
**          others                      = 2.
*    READ TABLE resulting_idocs INDEX 1.
*    IF sy-subrc = 0.
*      idoc_number = resulting_idocs-docnum.
*    ELSE.
*      CLEAR idoc_number.
*    ENDIF.
*  ENDIF.

 SORT et_resulting_idocs.
 DELETE ADJACENT DUPLICATES FROM et_resulting_idocs.

ENDFUNCTION.
>>>
```


Configuration Exchange Infrastructure

In the following the settings in Integration Directory will be described:

From MDM to ECC (initial load) and from MDM to BPM (operational mode):

Display Receiver Determination | SAP_MDM_EP_DEV | ABS_MI_MDM_CustomerMessage — Active

Type of Receiver Determination

Standard Extended

Configured Receivers

Condition	Party	Service
(/CustomersMsg/Customers/ECC_System/ECC_System = INI)		SAP_ERP_EP_CT_UNT
(/CustomersMsg/Customers/ECC_System/ECC_System ≠ INI)		IP_MDM_Customer_to_SAP

If No Receiver Is Found, Proceed as Follows:

Terminate Message Processing with Error (Restart Possible)

End Message Processing Without Error (Restart not Possible)

Continue Message Processing with the Following Receiver: Party Service

Configuration Overview for Receiver Determination

Receiver (Partner | Service)	Interface Mapping	Receiver Agreement (Communication Channel)
| SAP_ERP_EP_CT_UNT		
DEBMAS.DEBMAS06	IM_MDM_Customers_SAP_To_DEBMAS06	CC_In_IDoc_SAP_I2P_MDM_Receiver
ADRMAS.ADRMAS02	IM_MDM_Customers_SAP_To_ADRMAS02	CC_In_IDoc_SAP_I2P_MDM_Receiver
| IP_MDM_Customer_to_SAP		
ABS_MI_MDM_CustomerMessage	IM_MDM_Customers_to_Customers_default	Not Required

From BPM to ECC (operational mode):

Display Receiver Determination | IP_MDM_Vendor_to_SAP | ABS_MI_MDM_VendorMessage

Type of Receiver Determination

Standard Extended

Configured Receivers

Condition	Party	Service
		SAP_ERP_EP_CT_UNT

If No Receiver Is Found, Proceed as Follows:

Terminate Message Processing with Error (Restart Possible)

End Message Processing Without Error (Restart not Possible)

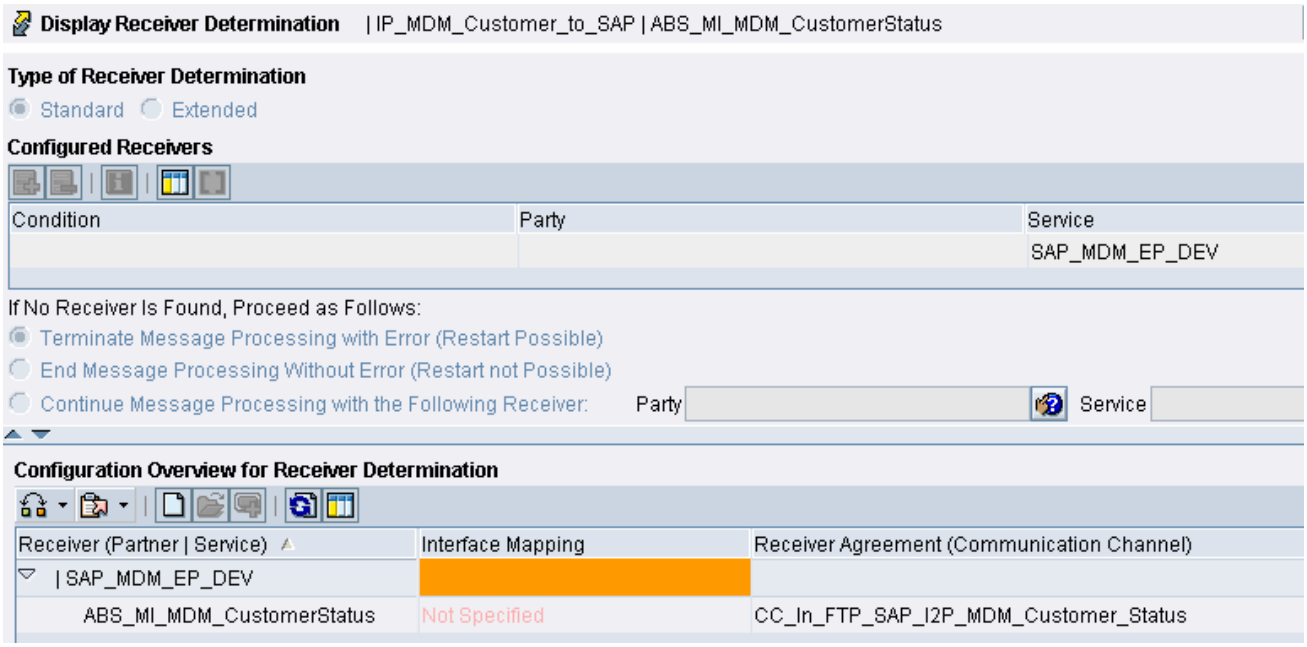
Continue Message Processing with the Following Receiver: Party Service

Configuration Overview for Receiver Determination

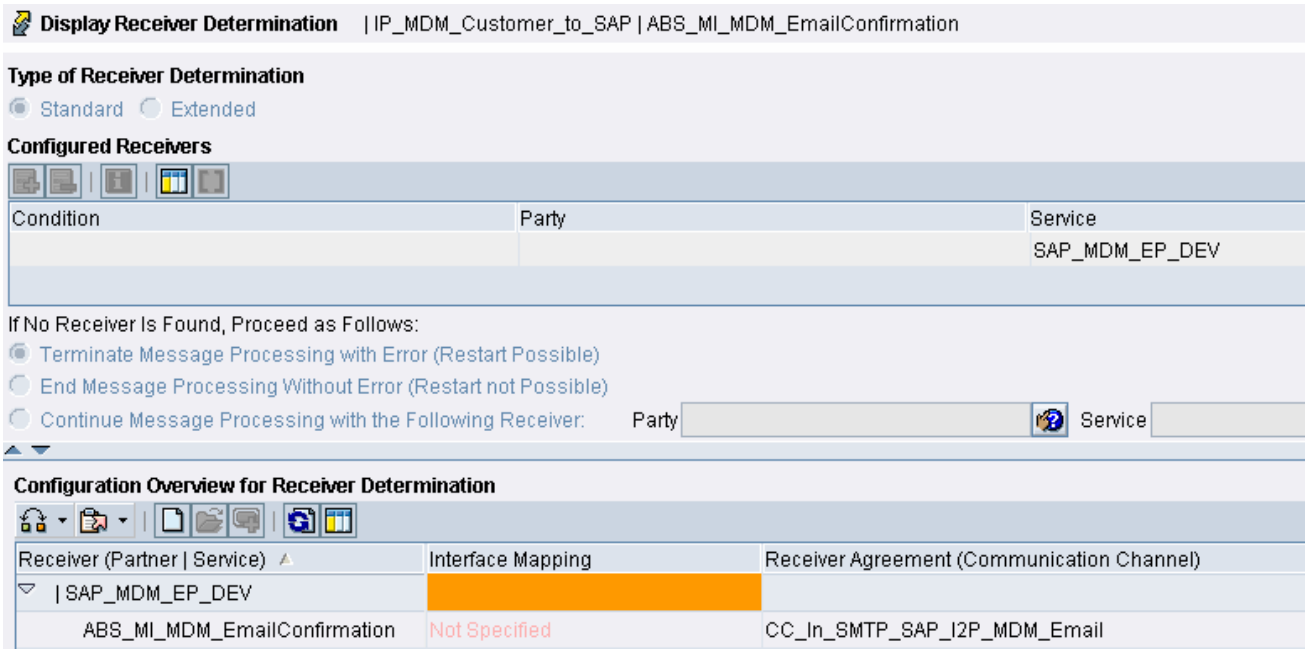
Receiver (Partner | Service)	Interface Mapping	Receiver Agreement (Communication Channel)
| SAP_ERP_EP_CT_UNT		
ADRMAS.ADRMAS02	IM_MDM_Vendors_to_SAP_ADRMAS02	CC_In_IDoc_SAP_I2P_MDM_Receiver
CREMAS.CREMAS04	IM_MDM_Vendors_to_SAP_CREMAS04	CC_In_IDoc_SAP_I2P_MDM_Receiver

The picture above shows distribution to only one backend system. If there will be more used, a separate condition for each system can be inserted with the designated service.

Confirmation Status from BPM to MDM:



Email from BPM to MDM (requestor and administrator):



SAP ECC customizing

Logical Systems

Within Transaction SALE the general ALE customizing for MDM vendor, customer and bank scenario for logical systems has to be defined. As we do have two different scenarios (initial load and operational mode) we need different logical systems. One for initial load which should be called MDMEPPRT (has to be defined also in the SLD with same name) and two for the BPM's. For customer it will be called MDMCUS and for vendor MDMVEN. Those settings have to be transported to all required ECC systems. Afterwards the distribution models have to be defined in BD64 and the ports might be adapted in WE20. For further information please refer to standard ALE customizing available in help.sap.com

Report variants for confirmation

In TC SE38 create a variant e.g. ZMDMAUDIT_100 for report ZRBDSTATE with the following parameters:

Confirmations to system: MDMCUS, MDMVEN

Message type: DEBMAS, CREMAS.

Do not specify a time period, the report will remember when it ran the last time and select the period accordingly. A new variant has to be created for each client

Schedule reports for confirmation

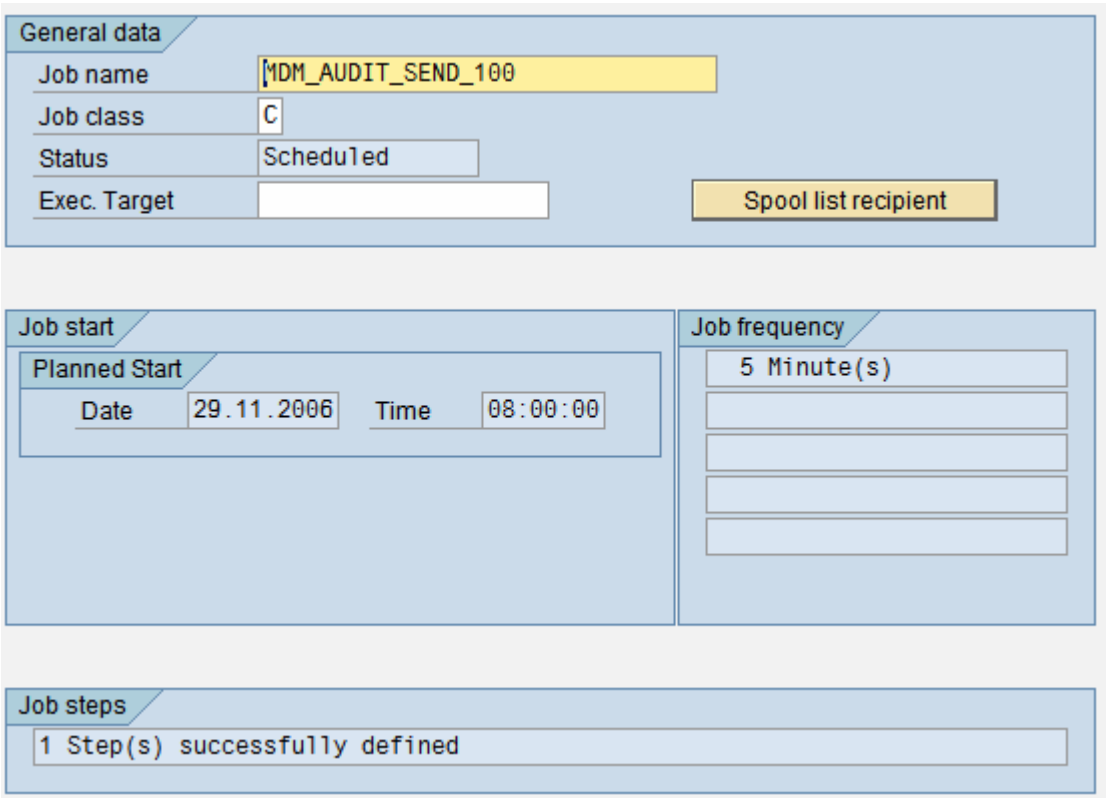
In TC SM36 schedule a report e.g. MDM_AUDIT_SEND_100 for each client.

Add a step with the parameters created earlier:

Name: ZRBDSTATE

Variant: ZMDMAUDIT_100

In the next step, a start condition needs to be created. A good value for the periodic rerun is 5 minutes.



MDM specific settings to repository

This document will primarily handle the configuration in XI and ECC but it should at least point to the right direction what is needed in MDM.

For various reasons, we decided to do the complex mappings outside of MDM in XI and having the syndication map only exporting the repository structure to an XSD 1:1. This XSD will then be imported into XI and all mappings are done there.

First of all, some fields for confirmation have to be added to the repository. As we want to be able distributing to multiple backend systems, an appropriate structure need to be created within the repository.

We created a lookup qualified flat “Confirmations” with the following fields:

	Pos.	Name	Type	Keyword	DF	UF	Qualifier
▶	[1]	SAPECC	Lookup [Flat]	None	[1]		No
	[2]	Status	Text [8]	None	[2]		Yes
	[3]	Objkey	Text [20]	None	[3]		Yes

When using multiple backend systems and if you want to decide in MDM if it will be distributed to all or dedicated systems, then you would need an additional field e.g. “Ecc_System” which could hold “all”, “individual1” and“individual2” etc. Then to have the process more flexible, another field “backendSystemCount” could be added to the repository. This field will tell the integration process later on with the value mapping variable how many loops are needed (see design section “customer”).

To be able to send an email to an individual user (creator of data set in MDM), one more field “requestorEmail” is required in MDM repository.

As the integration process is build, it expects single files for data distribution. However, it should also be possible to develop a business process where multi line files are possible.

Once the data has been syndicated to the folder XI will pick it up and start the process flow.

Importing the confirmation from XI

Once the customer was posted in ECC and the integration process has mapped all data and sent the confirmation back to MDM file system (inbound port of repository). The file needs to be imported via import manager. In our case, the message had the following format:


```
<?xml version="1.0" encoding="UTF-8"?>
<CustomerMsg>
  <Customers>
    <Customer_Number>0000000070</Customer_Number>
    <Old_Customer_Number>012341234</Old_Customer_Number>
    <Name_1>Steffen</Name_1>
    <Street>Althardstrasse</Street>
    <City>Regensdorf</City>
    <VAT_Registration_Number>CH123456789</VAT_Registration_Number>
    <PO_Box_City>8052</PO_Box_City>
    <Confirmation>
      <SAPECC>
        <SAPECC>DR0200</SAPECC>
      </SAPECC>
      <Status>53</Status>
      <Objkey>0000000070</Objkey>
    </Confirmation>
    <Confirmation>
      <SAPECC>
        <SAPECC>DR0100</SAPECC>
      </SAPECC>
      <Status>53</Status>
      <Objkey>0000000070</Objkey>
    </Confirmation>
  </Customers>
</CustomerMsg>
```

We've got some "header" information including customer number, name, street etc. to do the matching against the existing data in the repository, and the confirmation information from the backend system. In our case, the customer number in each system (when successfully posted) is visible in the portal or data manager. Status "53" reflects the IDoc status of the backend system.

Related Content

Please find more information on Exchange Infrastructure and the standard ALE customizing on the SAP help portal and SDN.

Exchange Infrastructure

SDN: Exchange Infrastructure

SDN: Master Data Management

Copyright