How To... Implement a Transformation End Routine

Applicable Releases:
SAP NetWeaver BI 3.x and 7.0

IT Practice:
Business Information Management

IT Scenario:
Enterprise Data Warehouse

Version 1.2
November 2008
## Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.20</td>
<td>Addition of SAP Note #1223532 as prerequisite.</td>
</tr>
<tr>
<td>1.10</td>
<td>Reference to HowTo paper about disaggregation on upload (Return Table feature).</td>
</tr>
<tr>
<td>1.00</td>
<td>First official release of this guide</td>
</tr>
</tbody>
</table>
### Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example Text</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation</td>
</tr>
<tr>
<td><strong>Example Text</strong></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td><strong>Example Text</strong></td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><strong>Example Text</strong></td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><strong>&lt;Example text&gt;</strong></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

### Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td>Caution</td>
</tr>
<tr>
<td>🔄</td>
<td>Note or Important</td>
</tr>
<tr>
<td>🔄</td>
<td>Example</td>
</tr>
<tr>
<td>🔄</td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
1. Business Scenario

Your Source delivers the sales figures for the individual sales organization. However, the sales figures need to be displayed for each employee. This means the incoming data needs to distribute according the employees belonging to the sales organization. The information which employee belongs to a sales organization is stored in the employee master data.

**Important**

In scenarios with insertion of new records into the result table (RESULT_TABLE), and you want to use error handling, please incorporate the information included in the SAP note #1223532.

**Note**

The approach with a routine in the Transformations can also be used as a replacement of the 3.x feature of Return Tables. For an example of how to use the Return Table feature, please refer to the online documentation or the HowTo guide “How To... Disaggregate on Upload”. This can be found in the SDN at SAP NetWeaver Capabilities ➔ SAP How-to Guides ➔ SAP NetWeaver 7.0 How-to Guides ➔ Business Information Management [https://www.sdn.sap.com/irj/sdn/howtoguides](https://www.sdn.sap.com/irj/sdn/howtoguides).

2. Background Information

The DataSource delivers the sales figures amount and quantity for a sales organization, material and month. The following processing steps are required in the End Routine of the Transformation:

- Read, from the master data table the employee IDs that belong to the sales organization
- Read, from master data table the number of employees that belong to a sales organization
- Divide the key figures the number of employees that have been read from the master data table
- Create an entry in the return table for each employee
End Routine

Incoming Change Parameter of End Routine

<table>
<thead>
<tr>
<th>Month</th>
<th>Sales Org.</th>
<th>Material</th>
<th>Currency</th>
<th>Amount</th>
<th>Unit</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>200601</td>
<td>OST</td>
<td>0815</td>
<td>EUR</td>
<td>300000</td>
<td>PC</td>
<td>600</td>
</tr>
</tbody>
</table>

Incoming Change Parameter of End Routine

<table>
<thead>
<tr>
<th>Employee</th>
<th>Sales Org.</th>
</tr>
</thead>
<tbody>
<tr>
<td>25671</td>
<td>OST</td>
</tr>
<tr>
<td>25672</td>
<td>OST</td>
</tr>
<tr>
<td>25673</td>
<td>OST</td>
</tr>
</tbody>
</table>

Outgoing Change Parameter of End Routine

<table>
<thead>
<tr>
<th>Month</th>
<th>Sales Org.</th>
<th>Material</th>
<th>Employee</th>
<th>Currency</th>
<th>Amount</th>
<th>Unit</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>200601</td>
<td>OST</td>
<td>0815</td>
<td>25671</td>
<td>EUR</td>
<td>100000</td>
<td>PC</td>
<td>200</td>
</tr>
<tr>
<td>200601</td>
<td>OST</td>
<td>0815</td>
<td>25672</td>
<td>EUR</td>
<td>100000</td>
<td>PC</td>
<td>200</td>
</tr>
<tr>
<td>200601</td>
<td>OST</td>
<td>0815</td>
<td>25673</td>
<td>EUR</td>
<td>100000</td>
<td>PC</td>
<td>200</td>
</tr>
</tbody>
</table>

The number of entries in the result table corresponds to the number of employees read from the table.

Please have a look at the How-to paper “How To... Routines within Transformations” to get familiar with the routines concept within Transformations. It can be found in the “SAP Business Intelligence How-To Guides for SAP NetWeaver 7.0” in the SDN (http://www.sdn.sap.com/irj/sdn/howtogs).
3. Step-by-Step Procedure

3.1 Prerequisite

1. Create InfoCube: SALES12

   Structure of the InfoCube:
   - 0CALMONTH
   - 0SALEDSORG
   - 0MATERIAL
   - 0CURRENCY
   - 0AMOUNT
   - 0UNIT
   - 0QUANTITY

2. Create DataSource: SALESDATA

   Structure of the DataSource:
   - 0CALMONTH
   - 0SALESORG
   - 0MATERIAL
   - 0CURRENCY
   - 0AMOUNT
   - 0UNIT
   - 0QUANTITY
3. **Enhance Employee Master Data (0EMPLOYEE) with Sales Organization (0SALES_ORG)**

   The sales organization is a Attribute of the Employee master data.

4. **Maintain the Sales Organization for the Employees**

5. **Create Transformation between the DataSource SALESDATA and the InfoCube SALES123**
6. Create Data Transfer Process between the DataSource SALESDATA and InfoCube SALES12

7. Create a Flat File (csv) corresponding to the DataSource structure in 2.

8. Create an InfoPackage corresponding to the files created in 7.
### 3.2 Create End Routine

1. Change Transformation between DataSource SALESDATA and InfoCube SALES12

   ![Transformation Process Image]

2. Create End Routine

   ![Transformation Change Image]

3. Copy the Coding into the End Routine. Please find the coding in the Appendix

4. Result

   ![List Output Image]
4. Appendix

Appendix A – Source Coding

In scenarios with insertion of new records into the result table (RESULT_TABLE), be aware of SAP note #1223532 for correct error handling.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>End Routine</td>
<td>Sample End Routine in the Transformation</td>
</tr>
</tbody>
</table>

PROGRAM trans_routine.

*---------------------------------------------------------------------*
*       CLASS routine DEFINITION                                      *
*---------------------------------------------------------------------*

CLASS routine DEFINITION.
PUBLIC SECTION.

TYPES:
BEGIN OF _ty_s_TG_1,
  * InfoObject: 0SALESORG Sales Organization.
    SALESORG TYPE /BI0/OISALESORG,
  * InfoObject: 0MATERIAL Material.
    MATERIAL TYPE /BI0/OIMATERIAL,
  * InfoObject: 0CALMONTH Calendar Year/Month.
    CALMONTH TYPE /BI0/OICALMONTH,
  * InfoObject: 0EMPLOYEE Employee.
    EMPLOYEE TYPE /BI0/OIEMPLOYEE,
  * InfoObject: 0AMOUNT Amount.
    AMOUNT TYPE /BI0/OIAMOUNT,
  * InfoObject: 0QUANTITY Quantity.
    QUANTITY TYPE /BI0/OIQUANTITY,
  * InfoObject: 0CURRENCY Currency Key.
    CURRENCY TYPE /BI0/OICURRENCY,
  * InfoObject: 0UNIT Unit of Measure.
    UNIT TYPE /BI0/OIUNIT,
  * Field: RECORD.
    RECORD TYPE RSARECORD,
END OF _ty_s_TG_1.

TYPES:_ty_t_TG_1 TYPE STANDARD TABLE OF _ty_s_TG_1
WITH NON-UNIQUE DEFAULT KEY.

PRIVATE SECTION.

TYPE-POOLS: rsd, rstr.

'S*$ begin of global - insert your declaration only below this line ‘-‘

* Global data declaration
*****************************************************

* List of all Employees and corresponding sales organisation
DATA: BEGIN OF I_S_EMPLOYEE,
  SALESORG TYPE /BI0/MEMPLOYEE-SALESORG,
  EMPLOYEE TYPE /BI0/MEMPLOYEE-EMPLOYEE,
END OF I_S_EMPLOYEE.

DATA: i_t_employee like table of I_S_EMPLOYEE.
* List of the all sales organisations and nb. of employees in

November 2008
* the sales organisation
DATA: BEGIN OF I_S_SALESORG_COUNT,
  SALESORG TYPE /BI0/MEMPLOYEE-SALESORG,
  counter TYPE i,
END OF I_S_SALESORG_COUNT.
DATA: i_t_SALESORG_COUNT like table of I_S_SALESORG_COUNT.

** end of global - insert your declaration only before this line **
METHODS
end_routine
IMPORTING
  request                  type rsrequest
  datapackid               type rstr_datapid
EXPORTING
  monitor                  type rstr_ty_t_monitors
CHANGING
  RESULT_PACKAGE              type _ty_t_TG_1
RAISING
  cx_rsrout_abort.
ENDCLASS.                    "routine DEFINITION

*---------------------------------------------------------------------*
*       CLASS routine IMPLEMENTATION
*---------------------------------------------------------------------*
*
*----------------------------------------------------------------------*
*       Method end_routine
*----------------------------------------------------------------------*
* Calculation of result package via end routine
* ----------------------------------------------------------------------*
* <-> result package
*----------------------------------------------------------------------*
METHOD end_routine.
*=== Segments ===
FIELD-SYMBOLS:
  <RESULT_FIELDS>    TYPE _ty_s_TG_1.
DATA:
  MONITOR_REC     TYPE rstmonitor.

** begin of routine - insert your code only below this line **
*****************************************************************************
* local data declaration
*****************************************************************************
data: e_s_result type _ty_s_TG_1.
data: e_t_result type _ty_t_TG_1.
data: amount   like e_s_result-amount.
data: quantity like e_s_result-quantity.
*****************************************************************************
* read master data in local table once
*****************************************************************************
  read table i_t_employee index 1 transporting no fields.
    if sy-subrc = 4.
  * get master data for all employees of the RESULT_PACKAGE
    select employee salesorg from /BI0/MEMPLOYEE
    into corresponding fields of table i_t_employee

November 2008
where OBJVERS = 'A'
  and DATETO >= sy-datum
  and DATEFROM <= sy-datum
  and salesorg <> '``'.
if sy-subrc = 4.
  RAISE EXCEPTION TYPE CX_RSROUT_ABORT.
endif.

sort i_t_employee by salesorg.

* select nb. of employees of a sales organisation
select salesorg COUNT( * ) as counter
from /BI0/MEMPLOYEE
  into corresponding fields of table i_t_salesorg_count
where OBJVERS = 'A'
  and DATETO >= sy-datum
  and DATEFROM <= sy-datum
  and salesorg <> '``'
  group by salesorg
  order by salesorg.
if sy-subrc = 4.
  RAISE EXCEPTION TYPE CX_RSROUT_ABORT.
endif.
endif.

***************************************************
*   Do the calculation and add the new rows
***************************************************

* loop over the input result data package
loop at RESULT_PACKAGE into e_s_result.
  * get Counter
read table i_t_salesorg_count into i_s_salesorg_count with key
  salesorg = e_s_result-salesorg.
  * if no counter is available input = output
  if sy-subrc <> 0.
    append e_s_result to e_t_result.
  * add the employees and devide the amount/quantity by counter
    else.
    " calculate quantity and amount
    if i_s_salesorg_count-counter <> 0.
      amount   = e_s_result-amount   /
      i_s_salesorg_count-counter.
      quantity = e_s_result-quantity   /
      i_s_salesorg_count-counter.
    else.
      " counter is 0
      RAISE EXCEPTION TYPE CX_RSROUT_ABORT.
    endif.
    loop at i_t_employee into i_s_employee where
      salesorg = e_s_result-salesorg.
      move i_s_employee-employee to e_s_result-employee.
      move amount to e_s_result-amount.
      move quantity to e_s_result-quantity.
    * append the added lines to result
    append e_s_result to e_t_result.
  * add the lines to the output package
endloop.
endif.
endloop.

*$*$ end of routine - insert your code only before this line
ENDMETHOD.        "end_routine
ENDCLASS.          "routine IMPLEMENTATION