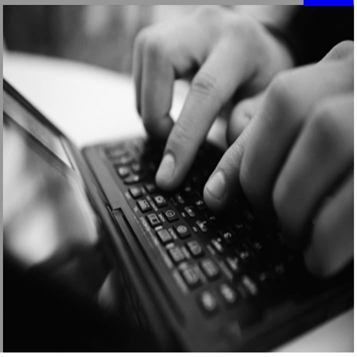
How-to Guide
SAP NetWeaver 2004s



# How To… Routines within Transformations

Version 1.00 – February 2006

Applicable Releases:
SAP NetWeaver 2004s
(BI Capabilities of SAP NetWeaver 2004s)

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.
SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.
SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.
SAP NetWeaver “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.
Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

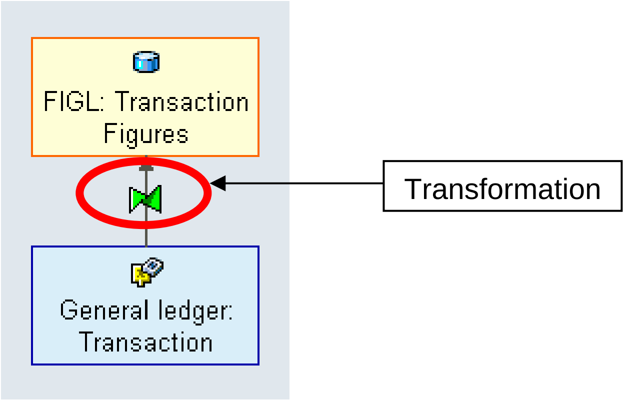
# 1 Business Scenario

You have been tasked to manipulate and transform the data flowing through your data warehouse. You were asked to do the following:

1) You need to delete a select number of records within the data package based upon a particular pattern that the records being loaded have:
2) Populate a field based upon values of other fields being passed in the load process.
3) Derive an additional field for the target record structure

In order to accomplish these tasks it has been determined that you need to invoke the use of routines within the transformations of SAP NetWeaver 2004s Business Intelligence.

For purposes of our example, (use of these routines is of course not limited to this particular example) your company is implementing FI-GL and loading data from the FI_GL_1 DataSource in SAP R/3. You are loading this data into the level 1 DataStore Object 0FIGL_O06. See diagram below:



This transformation will be enhanced via routines by performing the following actions:

1) All records that do not have either a value for Debit Postings or Credit Postings will be deleted from the data package in the start routine
2) The Debit/Credit Indicator field in the target structure will be populated in an individual characteristic routine.
3) An additional Plan/Actual field will be populated in the end routine.

# 2 Introduction

Within the BI Capabilities for SAP NetWeaver 2004s there is a new methodology for manipulating data as it moves through your SAP Business Information Warehouse. This methodology is called Transformations.

Transformations are made up at least **one** transformation rule.

- Transformation rules map any number of source fields to at least one field in the target. You can use different rules types for this.
- The different rule types available within a transformation rule are laid out below
    - Constants – A field can be filled directly with a value that is predetermined
    - Formula – Formula Builder and Transformation Library
    - Reading Master Data – Derive values off of existing master data
    - Time Update – When performing a time update, an automatic time conversion and a time distribution are available
    - Unit of Measure Conversion / Currency Translation
    - Initial – Leave a fields value blank
    - **Routines** – Custom code which you have written that will determine based on the programmed logic what the eventual value(s) of a field will be.

In this how-to paper we will be focus on how these routines can be used within our data warehouse to meet different needs.

The different types of routines that are available within Transformations are listed below:

- **The Start Routine**
- Routine for updating Key Figures
- **Routine for updating Characteristics**
- **End Routine**
- Expert Routine

The three transformation routine types high lighted above will be those which we will address today in our scenario.

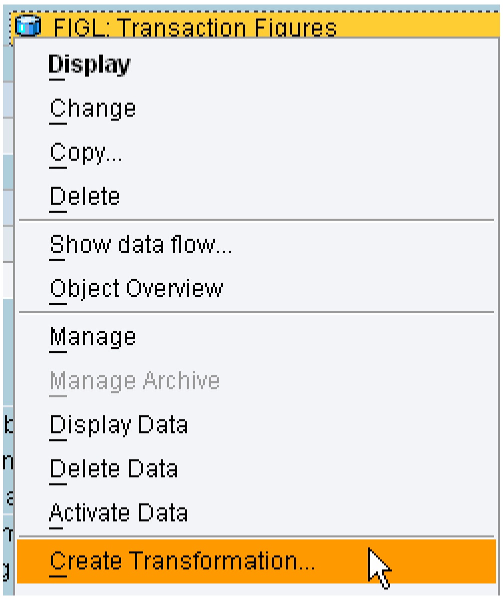
**NOTE:** Routines as they existed in previous releases are no longer subroutines, they are ABAP Objects methods. Similar to the migration from procedural to object-oriented programming seen with the move from CMOD Customer Exits to BaDI's in SAP BI we are seeing this same move here with the transition from Update/Transfer Rules to Transformations.

# 3 Step by Step Solution

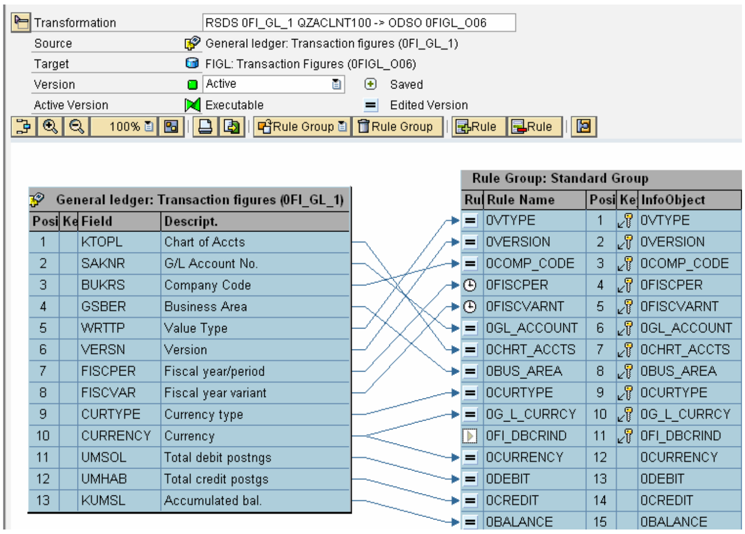
In order to eliminate all zero debit and credit records coming through in the data package a transformation is needed.

## 3.1 Create a Transformation

1. Right Click on the target object and select the Create Transformation option.



2. Create the relevant direct mappings by dragging and dropping the source field to their relevant targets.



| | | |
|---|---|---|
| Transformation | | RSDS 0FI_GL_1 QZACLNT100 -> ODSO 0FIGL_O06 |
| Source | | General ledger: Transaction figures (0FI_GL_1) |
| Target | | FIGL: Transaction Figures (0FIGL_O06) |
| Version | Active | Saved |
| Active Version | Executable | Edited Version |

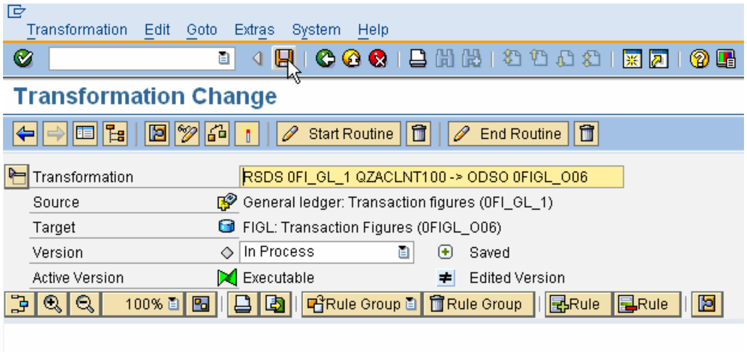
General ledger: Transaction figures (0FI_GL_1)

| Posi | Ke | Field | Descript. |
|---|---|---|---|
| 1 | | KTOPL | Chart of Accts |
| 2 | | SAKNR | G/L Account No. |
| 3 | | BUKRS | Company Code |
| 4 | | GSBER | Business Area |
| 5 | | WRTTP | Value Type |
| 6 | | VERSN | Version |
| 7 | | FISCPER | Fiscal year/period |
| 8 | | FISCVAR | Fiscal year variant |
| 9 | | CURTYPE | Currency type |
| 10 | | CURRENCY | Currency |
| 11 | | UMSOL | Total debit postngs |
| 12 | | UMHAB | Total credit postgs |
| 13 | | KUMSL | Accumulated bal. |

Rule Group: Standard Group

| Ru | Rule Name | Posi | Ke | InfoObject |
|---|---|---|---|---|
| = | 0VTYPE | 1 | ⚷ | 0VTYPE |
| = | 0VERSION | 2 | ⚷ | 0VERSION |
| = | 0COMP_CODE | 3 | ⚷ | 0COMP_CODE |
| ◷ | 0FISCPER | 4 | ⚷ | 0FISCPER |
| ◷ | 0FISCVARNT | 5 | ⚷ | 0FISCVARNT |
| = | 0GL_ACCOUNT | 6 | ⚷ | 0GL_ACCOUNT |
| = | 0CHRT_ACCTS | 7 | ⚷ | 0CHRT_ACCTS |
| = | 0BUS_AREA | 8 | ⚷ | 0BUS_AREA |
| = | 0CURTYPE | 9 | ⚷ | 0CURTYPE |
| = | 0G_L_CURRCY | 10 | ⚷ | 0G_L_CURRCY |
| ▷ | 0FI_DBCRIND | 11 | ⚷ | 0FI_DBCRIND |
| = | 0CURRENCY | 12 | | 0CURRENCY |
| = | 0DEBIT | 13 | | 0DEBIT |
| = | 0CREDIT | 14 | | 0CREDIT |
| = | 0BALANCE | 15 | | 0BALANCE |

3. Now save the transformation rule group.



Transformation: RSDS 0FI_GL_1 QZACLNT100 -> ODSO 0FIGL_O06

Source: General ledger: Transaction figures (0FI_GL_1)

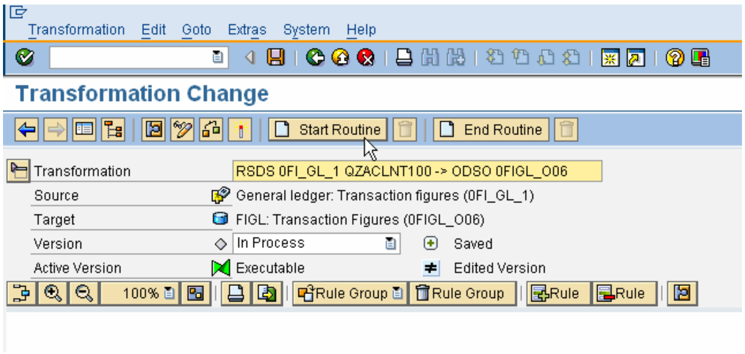
Target: FIGL: Transaction Figures (0FIGL_O06)

Version: In Process — Saved

Active Version: Executable — Edited Version

## 3.2 Create a Start Routine

1. From change mode in the Transformation click on the Create start routine button.



Transformation: RSDS 0FI_GL_1 QZACLNT100 -> ODSO 0FIGL_O06

Source: General ledger: Transaction figures (0FI_GL_1)

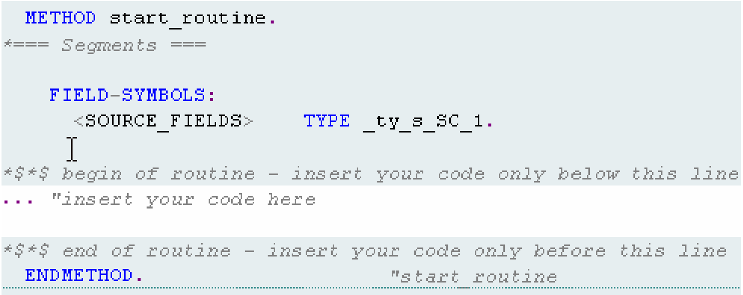
Target: FIGL: Transaction Figures (0FIGL_O06)

Version: In Process — Saved

Active Version: Executable — Edited Version

2. From within the start routine there are two sections of code to be filled.
   a. Global Section (optional)
   b. Local Section

   Navigate to the Local section



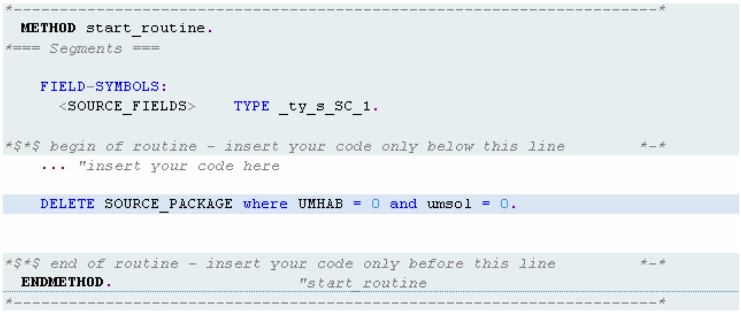
```
  METHOD start_routine.
*=== Segments ===

    FIELD-SYMBOLS:
      <SOURCE_FIELDS>    TYPE _ty_s_SC_1.

*$*$ begin of routine - insert your code only below this line
...  "insert your code here

*$*$ end of routine - insert your code only before this line
  ENDMETHOD.                    "start_routine
```

**NOTE:** Unlike within a subroutine where you would see the fields being passed into and out of the subroutine in the first line of the form declaration, within methods these fields are defined in the definition of the method within this class. (see appendix 4.1 for a detailed description of the start routine definition)

3. Tasked to eliminate all records that have neither a debit nor a credit, the first and only step is insert a delete statement

   - The source package is filtered.

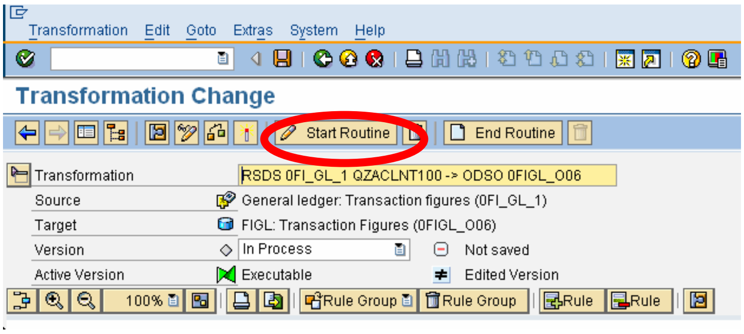
   The Start Routine is now complete



```
*---------------------------------------------------------------------*
  METHOD start_routine.
*=== Segments ===

    FIELD-SYMBOLS:
      <SOURCE_FIELDS>    TYPE _ty_s_SC_1.

*$*$ begin of routine - insert your code only below this line        *-*
    ... "insert your code here

    DELETE SOURCE_PACKAGE where UMHAB = 0 and umsol = 0.


*$*$ end of routine - insert your code only before this line         *-*
  ENDMETHOD.                    "start_routine
*---------------------------------------------------------------------*
```

4. Save the Start Routine and enter back into change mode for the Transformation. Save the Transformation as well.

   - There now exists a pencil on the start routine icon which is indicative of the fact that a start routine exists.



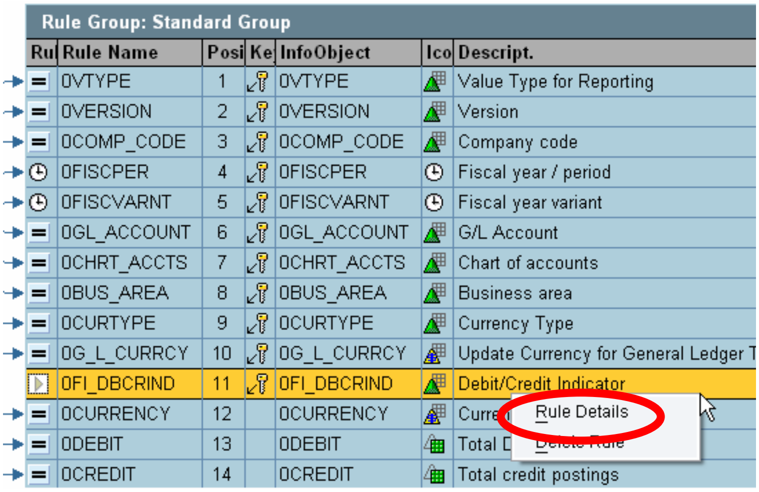
Transformation Change

| | | | |
|---|---|---|---|
| Transformation | RSDS 0FI_GL_1 QZACLNT100 -> ODSO 0FIGL_O06 | | |
| Source | General ledger: Transaction figures (0FI_GL_1) | | |
| Target | FIGL: Transaction Figures (0FIGL_O06) | | |
| Version | In Process | | Not saved |
| Active Version | Executable | | Edited Version |

## 3.3 Create a Routine for Updating Characteristics

1. As dictated by the customer the next step is to populate the Debit/Credit indicator with a value of ‘D’ if there is a debit posting on the record and a ‘C’ if there is a credit posting on the record.

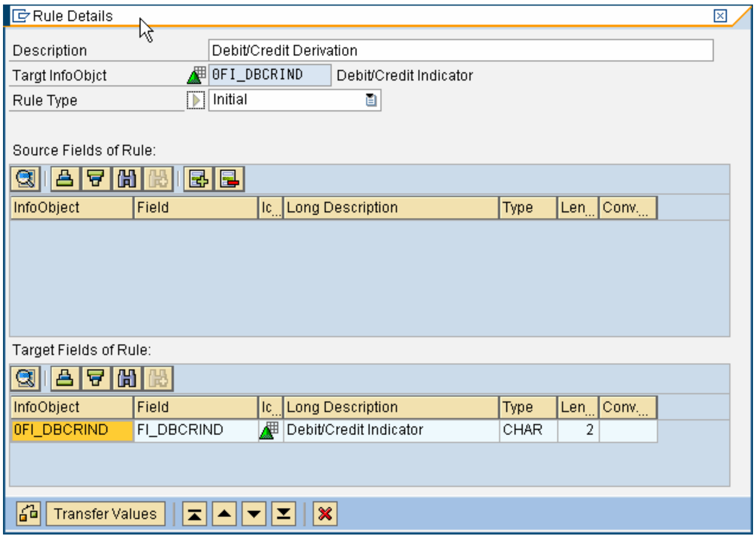
   - Right-Click on the Debit/Credit Indicator field within the rule group and click on the Rule Details button.



Rule Group: Standard Group

| Ru | Rule Name | Posi | Ke | InfoObject | Ico | Descript. |
|---|---|---|---|---|---|---|
| = | 0VTYPE | 1 | 🔑 | 0VTYPE | ▲ | Value Type for Reporting |
| = | 0VERSION | 2 | 🔑 | 0VERSION | ▲ | Version |
| = | 0COMP_CODE | 3 | 🔑 | 0COMP_CODE | ▲ | Company code |
| ◷ | 0FISCPER | 4 | 🔑 | 0FISCPER | ◷ | Fiscal year / period |
| ◷ | 0FISCVARNT | 5 | 🔑 | 0FISCVARNT | ◷ | Fiscal year variant |
| = | 0GL_ACCOUNT | 6 | 🔑 | 0GL_ACCOUNT | ▲ | G/L Account |
| = | 0CHRT_ACCTS | 7 | 🔑 | 0CHRT_ACCTS | ▲ | Chart of accounts |
| = | 0BUS_AREA | 8 | 🔑 | 0BUS_AREA | ▲ | Business area |
| = | 0CURTYPE | 9 | 🔑 | 0CURTYPE | ▲ | Currency Type |
| = | 0G_L_CURRCY | 10 | 🔑 | 0G_L_CURRCY | ▲ | Update Currency for General Ledger T |
| ▷ | 0FI_DBCRIND | 11 | 🔑 | 0FI_DBCRIND | ▲ | Debit/Credit Indicator |
| = | 0CURRENCY | 12 | | 0CURRENCY | ▲ | Curre |
| = | 0DEBIT | 13 | | 0DEBIT | ▦ | Total D |
| = | 0CREDIT | 14 | | 0CREDIT | ▦ | Total credit postings |

2. Give a Description to the rule being created.



Rule Details

Description: Debit/Credit Derivation

Targt InfoObjct: 0FI_DBCRIND Debit/Credit Indicator

Rule Type: Initial

Source Fields of Rule:

| InfoObject | Field | Ic.. | Long Description | Type | Len.. | Conv.. |
|---|---|---|---|---|---|---|

Target Fields of Rule:

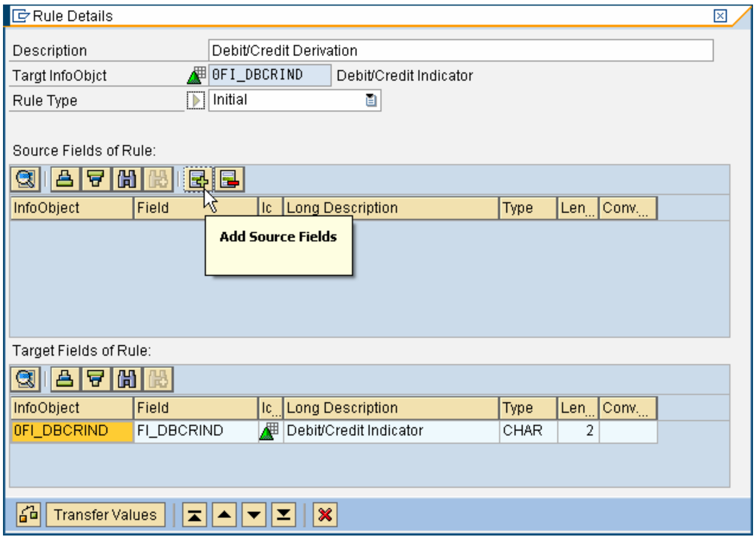
| InfoObject | Field | Ic.. | Long Description | Type | Len.. | Conv.. |
|---|---|---|---|---|---|---|
| 0FI_DBCRIND | FI_DBCRIND | | Debit/Credit Indicator | CHAR | 2 | |

3. Now the source fields:

UMSOL – *Total Debit Postgs*

UMHAB – *Total Credit Postgs*

need to be added to the rule so they can be accessed within the routine. Add the two fields and hit the Green OK button.



Rule Details

Description: Debit/Credit Derivation

Targt InfoObjct: 0FI_DBCRIND Debit/Credit Indicator

Rule Type: Initial

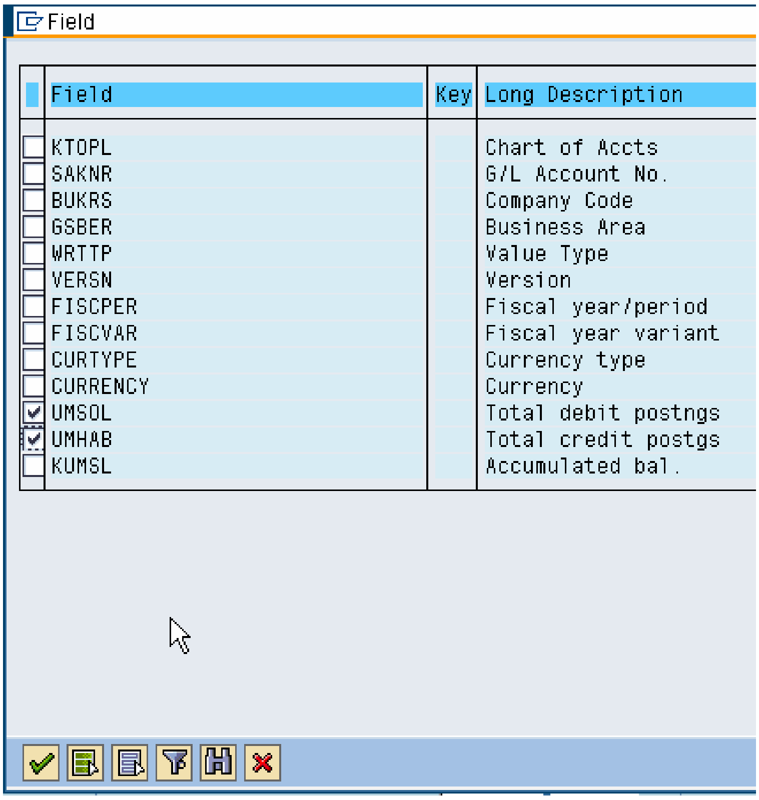
Source Fields of Rule:

| InfoObject | Field | Ic.. | Long Description | Type | Len.. | Conv.. |
|---|---|---|---|---|---|---|

Target Fields of Rule:

| InfoObject | Field | Ic.. | Long Description | Type | Len.. | Conv.. |
|---|---|---|---|---|---|---|
| 0FI_DBCRIND | FI_DBCRIND | | Debit/Credit Indicator | CHAR | 2 | |

**NOTE:** You could also assign the fields to the rule via drawing a link to the rule box in the netgraph UI of the transformations.

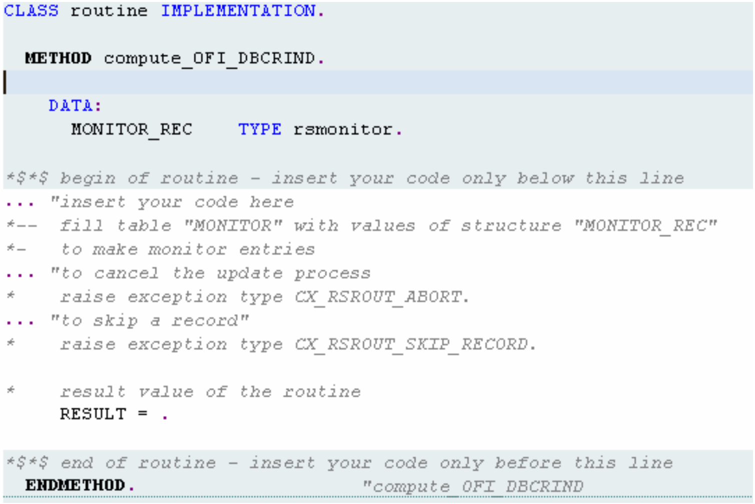


Field

|  | Field | Key | Long Description |
|---|---|---|---|
| □ | KTOPL |  | Chart of Accts |
| □ | SAKNR |  | G/L Account No. |
| □ | BUKRS |  | Company Code |
| □ | GSBER |  | Business Area |
| □ | WRTTP |  | Value Type |
| □ | VERSN |  | Version |
| □ | FISCPER |  | Fiscal year/period |
| □ | FISCVAR |  | Fiscal year variant |
| □ | CURTYPE |  | Currency type |
| □ | CURRENCY |  | Currency |
| ☑ | UMSOL |  | Total debit postngs |
| ☑ | UMHAB |  | Total credit postgs |
| □ | KUMSL |  | Accumulated bal. |

4. Within this piece of code the logic needs to be added to derive either a ‘D’ or a ‘C’ for our result field.

**NOTE:** Please see the appendix 4.2 for detailed description of the characteristic routine definition.

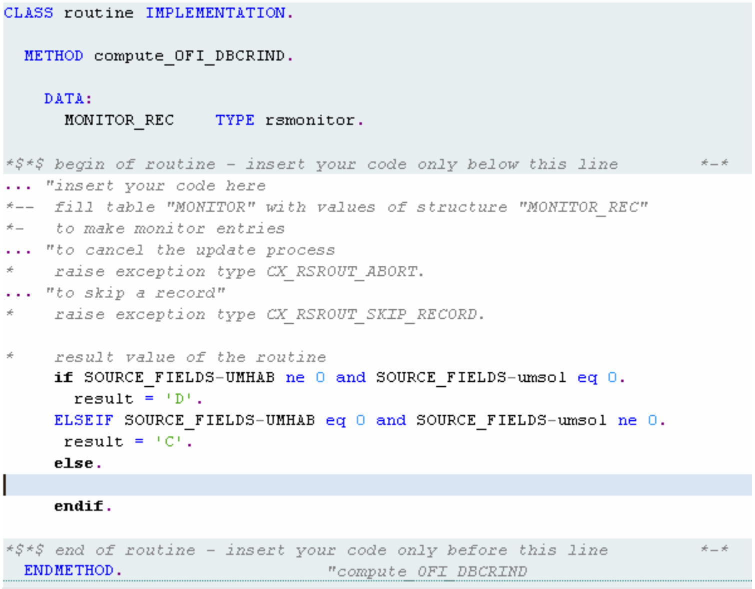


```
CLASS routine IMPLEMENTATION.

  METHOD compute_OFI_DBCRIND.

    DATA:
      MONITOR_REC     TYPE rsmonitor.

*$*$ begin of routine - insert your code only below this line
... "insert your code here
*--  fill table "MONITOR" with values of structure "MONITOR_REC"
*-   to make monitor entries
... "to cancel the update process
*    raise exception type CX_RSROUT_ABORT.
... "to skip a record"
*    raise exception type CX_RSROUT_SKIP_RECORD.

*    result value of the routine
     RESULT = .

*$*$ end of routine - insert your code only before this line
  ENDMETHOD.                    "compute_OFI_DBCRIND
```

5. A conditional statement needs to be created that determines whether the debit posting field has a value ( triggering the population of the the result with a ‘D’) or the credit posting field has a value (triggering the population of the result with a ‘C’).

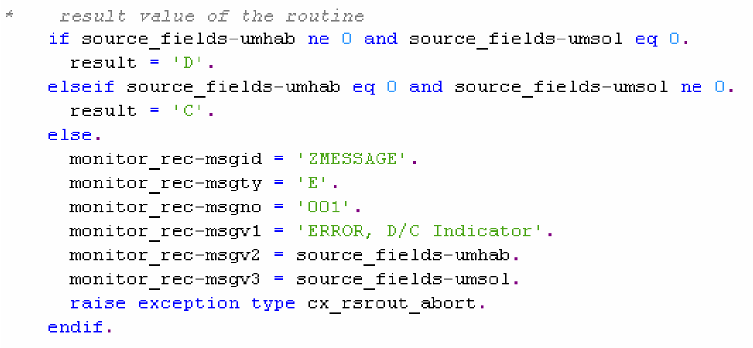
   - The debit and credit postings are checked for values if the debit posting has a value not equal to zero and the credit posting value is equal to zero we assign the value ‘D’, for debit to the debit/credit indicator.

   - On the other hand, if the credit posting’s value is not equal to zero and the debit posting’s value is, the value ‘C’, for credit is assigned to the debit/credit indicator.



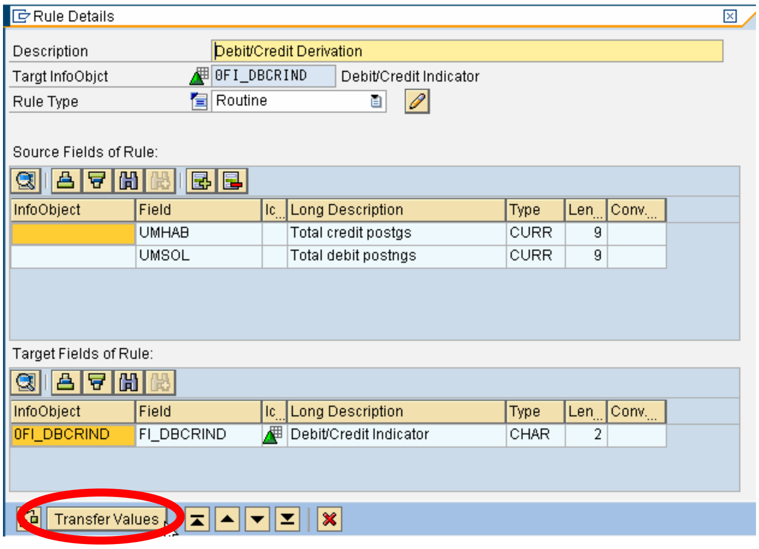
```
CLASS routine IMPLEMENTATION.

  METHOD compute_OFI_DBCRIND.

    DATA:
      MONITOR_REC     TYPE rsmonitor.

*$*$ begin of routine - insert your code only below this line        *-*
... "insert your code here
*--  fill table "MONITOR" with values of structure "MONITOR_REC"
*-   to make monitor entries
... "to cancel the update process
*    raise exception type CX_RSROUT_ABORT.
... "to skip a record"
*    raise exception type CX_RSROUT_SKIP_RECORD.

*    result value of the routine
     if SOURCE_FIELDS-UMHAB ne 0 and SOURCE_FIELDS-umsol eq 0.
       result = 'D'.
     ELSEIF SOURCE_FIELDS-UMHAB eq 0 and SOURCE_FIELDS-umsol ne 0.
      result = 'C'.
     else.

     endif.

*$*$ end of routine - insert your code only before this line         *-*
  ENDMETHOD.                    "compute_OFI_DBCRIND
```

6. The last step is to catch an exception if both the credit and debit fields have a value this is an error so a message needs to be written to the monitor and we will raise an exception to stop the load.



```
*     result value of the routine
    if source_fields-umhab ne 0 and source_fields-umsol eq 0.
      result = 'D'.
    elseif source_fields-umhab eq 0 and source_fields-umsol ne 0.
      result = 'C'.
    else.
      monitor_rec-msgid = 'ZMESSAGE'.
      monitor_rec-msgty = 'E'.
      monitor_rec-msgno = '001'.
      monitor_rec-msgv1 = 'ERROR, D/C Indicator'.
      monitor_rec-msgv2 = source_fields-umhab.
      monitor_rec-msgv3 = source_fields-umsol.
      raise exception type cx_rsrout_abort.
    endif.
```

7. Save the Characteristic Routine and Transfer the values back to the Rule Group. Save your Transformations.



| | | |
|---|---|---|
| Description | Debit/Credit Derivation | |
| Targt InfoObjct | 0FI_DBCRIND | Debit/Credit Indicator |
| Rule Type | Routine | |

Source Fields of Rule:

| InfoObject | Field | Ic... | Long Description | Type | Len... | Conv... |
|---|---|---|---|---|---|---|
| | UMHAB | | Total credit postgs | CURR | 9 | |
| | UMSOL | | Total debit postngs | CURR | 9 | |

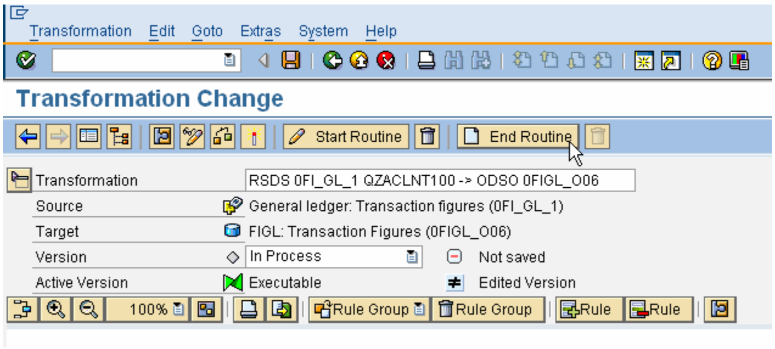
Target Fields of Rule:

| InfoObject | Field | Ic... | Long Description | Type | Len... | Conv... |
|---|---|---|---|---|---|---|
| 0FI_DBCRIND | FI_DBCRIND | | Debit/Credit Indicator | CHAR | 2 | |

## 3.4 Create End Routine

1. The final routine to be created is the end routine, this routine will populate the Plan/Actual Indicator. The routine will read the R/3 value type field and if the value being passed is a 10 (Actual), the value 'A' will be assigned to the Plan/Actual indicator. If the value type has the value 20 (Plan), the value 'P' will be assigned, otherwise the indicator will remain in its initial state.

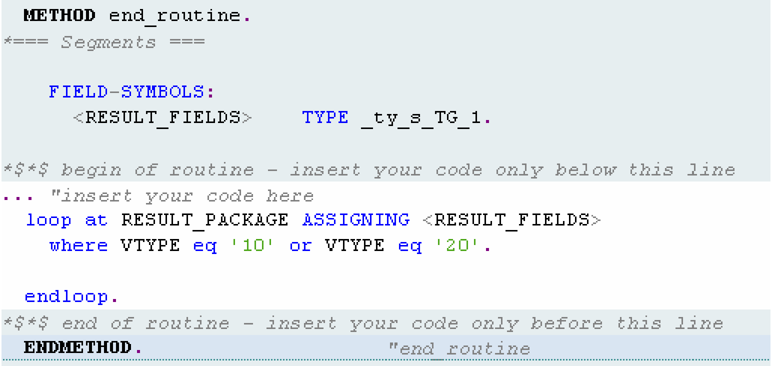
   Begin by clicking on the create button for the End Routine.



| | | | |
|---|---|---|---|
| Transformation | RSDS 0FI_GL_1 QZACLNT100 -> ODSO 0FIGL_O06 | | |
| Source | General ledger: Transaction figures (0FI_GL_1) | | |
| Target | FIGL: Transaction Figures (0FIGL_O06) | | |
| Version | In Process | | Not saved |
| Active Version | Executable | | Edited Version |

2. The end routine to be populated looks very similar to the start routine (see appendix 4.3 for details of the method definition's interface). The result_package needs to be looped through where the R/3 value types are either plan (20) or actual (10) value types.

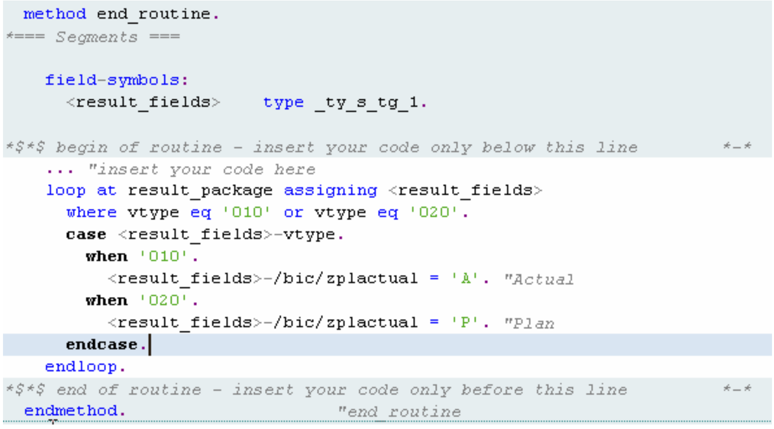
   - The code here is looping through the result_package into the field symbol <result_fields> provided by the method only for records that have the value types 10 or 20.



```
  METHOD end_routine.
*=== Segments ===

    FIELD-SYMBOLS:
      <RESULT_FIELDS>    TYPE _ty_s_TG_1.

*$*$ begin of routine - insert your code only below this line
... "insert your code here
  loop at RESULT_PACKAGE ASSIGNING <RESULT_FIELDS>
    where VTYPE eq '10' or VTYPE eq '20'.

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

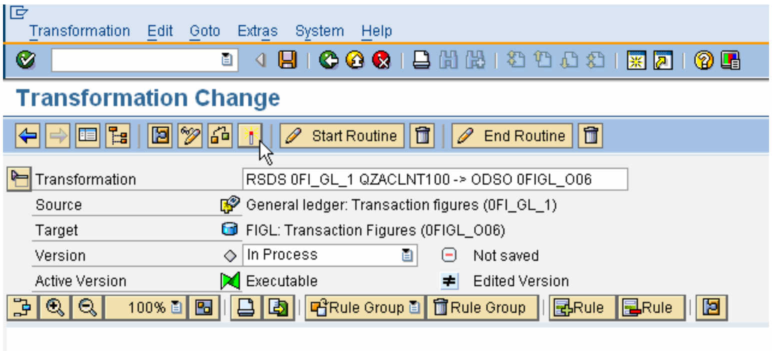
3. If the value type of a given record is 10 the plan/actual indicator receives the value ‘A’ for actual. If the value type is 20 the value passed to the plan/actual indicator is ‘P’ for plan.

    - The conditional case statement inserted evaluates the R/3 value type and based on its value gives the appropriate value to the plan/actual indicator.



```
 method end_routine.
*=== Segments ===

   field-symbols:
     <result_fields>    type _ty_s_tg_1.

*$*$ begin of routine - insert your code only below this line         *-*
   ... "insert your code here
   loop at result_package assigning <result_fields>
     where vtype eq '010' or vtype eq '020'.
     case <result_fields>-vtype.
       when '010'.
         <result_fields>-/bic/zplactual = 'A'. "Actual
       when '020'.
         <result_fields>-/bic/zplactual = 'P'. "Plan
     endcase.
   endloop.
*$*$ end of routine - insert your code only before this line          *-*
 endmethod.                    "end_routine
```

4. Save the end routine. Save and activate the Transformation.



Transformation Change

Transformation: RSDS 0FI_GL_1 QZACLNT100 -> ODSO 0FIGL_O06

Source: General ledger: Transaction figures (0FI_GL_1)

Target: FIGL: Transaction Figures (0FIGL_O06)

Version: In Process — Not saved

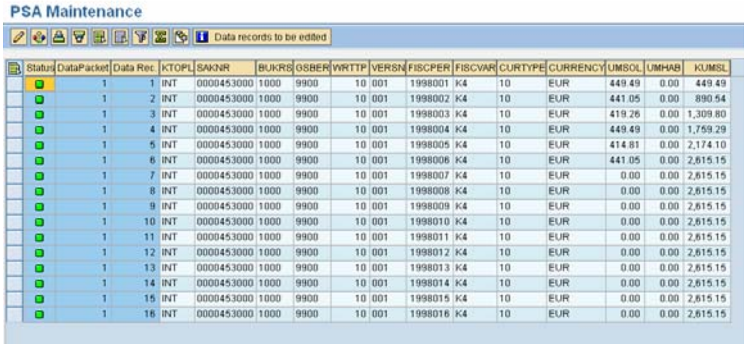
Active Version: Executable — Edited Version

## 3.5 Examine DataSource

1. Before starting the data load to test the new Transformation logic the dataSource should be examined.

    - The DataSource has been loaded with a limited set of records. Only records for G/L Account 0000453000 for the year 1998 have been loaded.

    - From the DataSource, it can be determined that the records for period 7 through 16 will be deleted. It can also be determined that all of the records are credit postings and are actuals due to the R/3 value type of 10 shown.



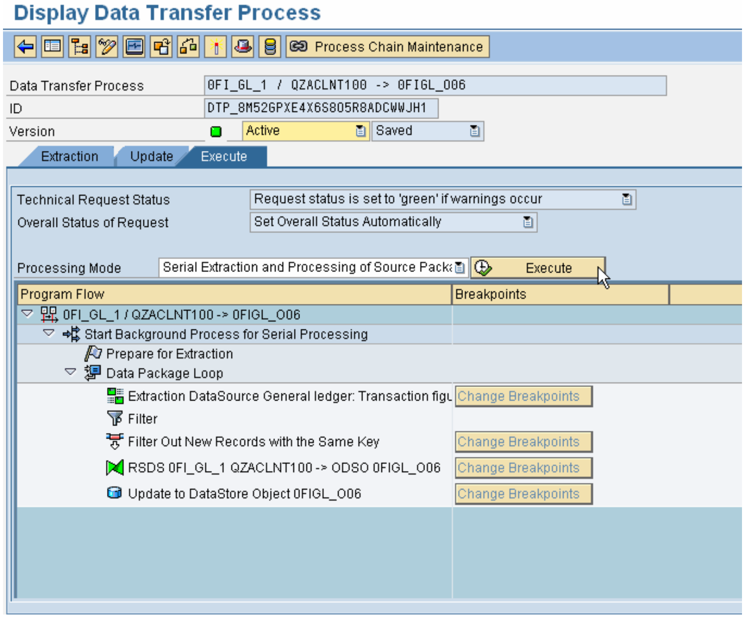
PSA Maintenance

| Status | DataPacket | Data Rec. | KTOPL | SAKNR | BUKRS | GSBER | WRTTP | VERSN | FISCPER | FISCVAR | CURTYPE | CURRENCY | UMSOL | UMHAB | KUMSL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 🟩 | 1 | 1 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998001 | K4 | 10 | EUR | 449.49 | 0.00 | 449.49 |
| 🟩 | 1 | 2 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998002 | K4 | 10 | EUR | 441.05 | 0.00 | 890.54 |
| 🟩 | 1 | 3 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998003 | K4 | 10 | EUR | 419.26 | 0.00 | 1,309.80 |
| 🟩 | 1 | 4 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998004 | K4 | 10 | EUR | 449.49 | 0.00 | 1,759.29 |
| 🟩 | 1 | 5 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998005 | K4 | 10 | EUR | 414.81 | 0.00 | 2,174.10 |
| 🟩 | 1 | 6 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998006 | K4 | 10 | EUR | 441.05 | 0.00 | 2,615.15 |
| 🟩 | 1 | 7 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998007 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |
| 🟩 | 1 | 8 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998008 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |
| 🟩 | 1 | 9 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998009 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |
| 🟩 | 1 | 10 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998010 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |
| 🟩 | 1 | 11 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998011 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |
| 🟩 | 1 | 12 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998012 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |
| 🟩 | 1 | 13 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998013 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |
| 🟩 | 1 | 14 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998014 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |
| 🟩 | 1 | 15 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998015 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |
| 🟩 | 1 | 16 | INT | 0000453000 | 1000 | 9900 | 10 | 001 | 1998016 | K4 | 10 | EUR | 0.00 | 0.00 | 2,615.15 |

## 3.6 Execute Data Transfer Process & Verify Results

1. The next step is to execute the data package.

   - From within the DataWarehousing Workbench identify the data transfer process created to load the 0FIGL_O06 DataStore Object right click and hit display.
   - Navigate to the execute tab.
   - Choose the execute button.



Display Data Transfer Process

Process Chain Maintenance

Data Transfer Process: 0FI_GL_1 / QZACLNT100 -> 0FIGL_O06
ID: DTP_8M526PXE4X6S805R8ADCWWJH1
Version: Active | Saved

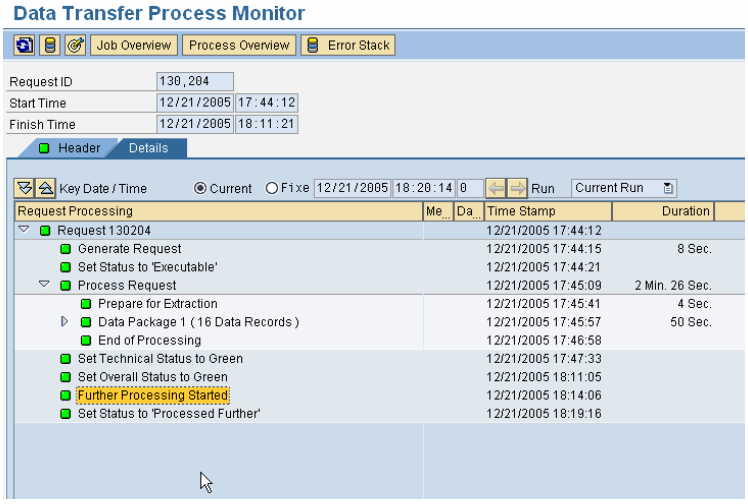
Extraction | Update | Execute

Technical Request Status: Request status is set to 'green' if warnings occur
Overall Status of Request: Set Overall Status Automatically

Processing Mode: Serial Extraction and Processing of Source Pack | Execute

| Program Flow | Breakpoints |
|---|---|
| 0FI_GL_1 / QZACLNT100 -> 0FIGL_O06 | |
| Start Background Process for Serial Processing | |
| Prepare for Extraction | |
| Data Package Loop | |
| Extraction DataSource General ledger: Transaction figu | Change Breakpoints |
| Filter | |
| Filter Out New Records with the Same Key | Change Breakpoints |
| RSDS 0FI_GL_1 QZACLNT100 -> ODSO 0FIGL_O06 | Change Breakpoints |
| Update to DataStore Object 0FIGL_O06 | Change Breakpoints |

2. Verify within the Data Transfer Process Monitor that the load into the DataStore Object 0FIGL_O06 was successful.



Data Transfer Process Monitor

Job Overview | Process Overview | Error Stack

Request ID: 130,204
Start Time: 12/21/2005 17:44:12
Finish Time: 12/21/2005 18:11:21

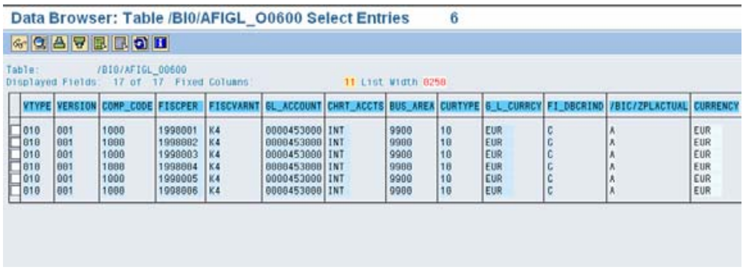
Header | Details

Key Date / Time ● Current ○ Fixe 12/21/2005 18:20:14 0 Run Current Run

| Request Processing | Me... | Da... | Time Stamp | Duration |
|---|---|---|---|---|
| 🟩 Request 130204 | | | 12/21/2005 17:44:12 | |
| 🟩 Generate Request | | | 12/21/2005 17:44:15 | 8 Sec. |
| 🟩 Set Status to 'Executable' | | | 12/21/2005 17:44:21 | |
| 🟩 Process Request | | | 12/21/2005 17:45:09 | 2 Min. 26 Sec. |
| 🟩 Prepare for Extraction | | | 12/21/2005 17:45:41 | 4 Sec. |
| 🟩 Data Package 1 ( 16 Data Records ) | | | 12/21/2005 17:45:57 | 50 Sec. |
| 🟩 End of Processing | | | 12/21/2005 17:46:58 | |
| 🟩 Set Technical Status to Green | | | 12/21/2005 17:47:33 | |
| 🟩 Set Overall Status to Green | | | 12/21/2005 18:11:05 | |
| 🟩 Further Processing Started | | | 12/21/2005 18:14:06 | |
| 🟩 Set Status to 'Processed Further' | | | 12/21/2005 18:19:16 | |

3. Validate that the data and the transformation logic performed as expected.

   - Six records should have loaded for the first six periods of 1998 for GL/Account 0000453000.
   - The six records should all be marked with a Debit/Credit indicator 'C'
   - All six records should also have a value of 'A' for the plan/actual indicator.



Data Browser: Table /BI0/AFIGL_O0600 Select Entries 6

Table: /BI0/AFIGL_O0600
Displayed Fields: 17 of 17 Fixed Columns: 11 List Width 0250

| | VTYPE | VERSION | COMP_CODE | FISCPER | FISCVARNT | GL_ACCOUNT | CHRT_ACCTS | BUS_AREA | CURTYPE | G_L_CURRCY | FI_DBCRIND | /BIC/ZPLACTUAL | CURRENCY |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| □ | 010 | 001 | 1000 | 1998001 | K4 | 0000453000 | INT | 9900 | 10 | EUR | C | A | EUR |
| □ | 010 | 001 | 1000 | 1998002 | K4 | 0000453000 | INT | 9900 | 10 | EUR | C | A | EUR |
| □ | 010 | 001 | 1000 | 1998003 | K4 | 0000453000 | INT | 9900 | 10 | EUR | C | A | EUR |
| □ | 010 | 001 | 1000 | 1998004 | K4 | 0000453000 | INT | 9900 | 10 | EUR | C | A | EUR |
| □ | 010 | 001 | 1000 | 1998005 | K4 | 0000453000 | INT | 9900 | 10 | EUR | C | A | EUR |
| □ | 010 | 001 | 1000 | 1998006 | K4 | 0000453000 | INT | 9900 | 10 | EUR | C | A | EUR |

# 4 Appendix A: Example Transformations Code

## 4.1 START_ROUTINE

**INTERFACE**

```
METHODS
  start_routine
    IMPORTING
      request                  type rsrequest
      datapackid               type rsdatapid
    EXPORTING
      monitor                  type rstr_ty_t_monitors
    CHANGING
      SOURCE_PACKAGE              type _ty_t_SC_1
    RAISING
      cx_rsrout_abort.
```

- REQUEST – Request ID of load in progress
- DATAPACKID – data package ID (i.e. 1, 2, 3)
- MONITOR – Messaging mechanism for Transformations, place informational and error messages inside the structure, it will then be displayed from within the monitor.
- SOURCE_PACKAGE – (formerly DATA_PACKAGE) Contains all of the data being passed into the Transformation from the specified source
- CX_RSROUT_ABORT – Class Exception when raised will cause the transformation process to halt.

## 4.2 ROUTINE FOR UPDATING CHARACTERISTICS

**INTERFACE**

```
METHODS
  compute_0FI_DBCRIND
    IMPORTING
      request                  type rsrequest
      datapackid               type rsdatapid
      SOURCE_FIELDS              type _ty_s_SC_1
    EXPORTING
      RESULT                   type _ty_s_TG_1-FI_DBCRIND
      monitor                  type rstr_ty_t_monitor
    RAISING
      cx_rsrout_abort
      cx_rsrout_skip_record.
```

- REQUEST – Request ID of load in progress
- DATAPACKID – data package ID (i.e. 1, 2, 3)
- SOURCE_FIELDS – Structure containing the values of the field(s) specified as source fields when creating the rule.
- RESULT – Parameter that receives the desired result value for the characteristic.
- MONITOR – Messaging mechanism for Transformations, place informational and error messages inside the structure, it will then be displayed from within the monitor.
- CX_RSROUT_ABORT – Class Exception when raised will cause the transformation process to halt.

- CX_RSROUT_SKIP_RECORD – Class Exception when raised will cause the transformation process to skip a record.

## 4.3 END ROUTINE

**INTERFACE**

```
    METHODS
      end_routine
        IMPORTING
          request                    type rsrequest
          datapackid                 type rsdatapid
        EXPORTING
          monitor                    type rstr_ty_t_monitors
        CHANGING
          RESULT_PACKAGE                type _ty_t_TG_1
        RAISING
          cx_rsrout_abort.
```

- REQUEST – Request ID of load in progress
- DATAPACKID – data package ID (i.e. 1, 2, 3)
- MONITOR – Messaging mechanism for Transformations, place informational and error messages inside the structure, it will then be displayed from within the monitor.
- RESULT_PACKAGE –Contains all of the data being passed out of the Transformation after the the different Rule Groups have been processed, transforming the data.
- CX_RSROUT_ABORT – Class Exception when raised will cause the transformation process to halt.

# 5 Appendix B: Examples of Routines SAP BW 3.x -> BI Capabilities of NetWeaver 2004s

| BW 3.X | BI Capabilities of SAP NetWeaver 2004s |
|---|---|
| PROGRAM UPDATE_ROUTINE.<br>*$*$ begin of global - insert<br>your declaration only below this<br>line  *-*<br>* TABLES: ...<br>* DATA:   ...<br>*$*$ end of global - insert your<br>declaration only before this<br>tion is new<br>URE.<br>URE<br>/BIC/CS80D_PU_C01.<br>TYPES:<br>RECNO   LIKE sy-tabix,<br>END OF DATA_PACKAGE_STRUCTURE.<br>DATA:<br>DATA_PACKAGE TYPE STANDARD<br>TABLE OF DATA_PACKAGE_STRUCTURE<br>WITH HEADER LINE<br>WITH NON-UNIQUE DEFAULT<br>KEY INITIAL SIZE 0.<br><br>FORM startup<br>TABLES   MONITOR STRUCTURE<br>RSMONITOR "user defined<br>monitoring<br>MONITOR_RECNO<br>STRUCTURE RSMONITORS "<br>monitoring with record n<br>DATA_PACKAGE<br>STRUCTURE DATA_PACKAGE<br>USING    RECORD_ALL LIKE SY-<br>TABIX<br>SOURCE_SYSTEM LIKE<br>RSUPDSIMULH-LOGSYS<br>CHANGING ABORT LIKE SY-SUBRC.<br>"set ABORT <> 0 to cancel update<br>*<br>*$*$ begin of routine - insert<br>your code only below this line<br>*-*<br>* fill the internal tables<br>"MONITOR" and/or<br>"MONITOR_RECNO",<br>* to make monitor entries | PROGRAM trans_routine.<br><br>*---------------------------------------------<br>-------------------------*<br>*       CLASS routine DEFINITION<br>*---------------------------------------------<br>-------------------------*<br>*<br>*---------------------------------------------<br>-------------------------*<br>CLASS routine DEFINITION.<br>PUBLIC SECTION.<br><br>TYPES:<br>BEGIN OF _ty_s_SC_1,<br>*      Field: KTOPL Chart of Accts.<br>KTOPL           TYPE C LENGTH 4,<br>*      Field: SAKNR G/L Account No..<br>SAKNR           TYPE C LENGTH 10,<br>*      Field: BUKRS Company Code.<br>BUKRS           TYPE C LENGTH 4,<br>*      Field: GSBER Business Area.<br>GSBER           TYPE C LENGTH 4,<br>*      Field: WRTTP Value Type.<br>WRTTP           TYPE N LENGTH 3,<br>*      Field: VERSN Version.<br>VERSN           TYPE C LENGTH 3,<br>*      Field: FISCPER Fiscal year/period.<br>FISCPER           TYPE N LENGTH 7,<br>*      Field: FISCVAR Fiscal year variant.<br>FISCVAR           TYPE C LENGTH 2,<br>*      Field: CURTYPE Currency type.<br>CURTYPE           TYPE C LENGTH 2,<br>*      Field: CURRENCY Currency.<br>CURRENCY           TYPE C LENGTH 5,<br>*      Field: UMSOL Total debit postngs.<br>UMSOL           TYPE P LENGTH 9 DECIM<br>ALS 2,<br>*      Field: UMHAB Total credit postgs.<br>UMHAB           TYPE P LENGTH 9 DECIM<br>ALS 2,<br>*      Field: KUMSL Accumulated bal..<br>KUMSL           TYPE P LENGTH 9 DECIM<br>ALS 2,<br>*      Field: RECORD Record Number.<br>RECORD           TYPE RSARECORD,<br>END   OF _ty_s_SC_1.<br>TYPES:<br>_ty_t_SC_1        TYPE STANDARD TABLE O<br>F _ty_s_SC_1 |

**Start Routines**

**Here we can see the global data area of the BW 3.X systems. These are simply variables created within the program outside of any forms (subroutines).**

```
* if abort is not equal zero,
the update process will be
canceled
  ABORT = 0.

*$*$ end of routine - insert
your code only before this line
*-*
*
ENDFORM.
```

```
UNIQUE DEFAULT KEY.
  PRIVATE SECTION.

    TYPE-POOLS: rsd, r

*$*$ begin of global -
 insert your declarati
  *-*
    ... "insert your c
*$*$ end of global -
 insert your declarati
e   *-*
    METHODS
      start_routine
        IMPORTING
          request
uest
          datapackid
apid
        EXPORTING
          monitor
ty_t_monitors
        CHANGING
          SOURCE_PACKA
y_t_SC_1
        RAISING
          cx_rsrout_ab
    METHODS
      inverse_start_ro
        IMPORTING
          I_R_SELSET_O
EF TO CL_RSMDS_SET
          i_th_fields_
ashed table
          i_r_universe_
EF TO CL_RSMDS_UNIVERSE
        CHANGING
          c_r_selset_inbound          TYPE R
EF TO CL_RSMDS_SET
          c_th_fields_inbound         type h
ashed table
          c_exact                     type r
s_bool.
ENDCLASS.                     "routine DEFINIT
ION

*---------------------------------------------
-------------------------*
*       CLASS routine IMPLEMENTATION
*---------------------------------------------
-------------------------*
*
*---------------------------------------------
-------------------------*
CLASS routine IMPLEMENTATION.

*---------------------------------------------
--------------------------*
*       Method start_routine
*---------------------------------------------
```

**The global data area of the new Transformations is a bit different; the data is technically created within the Private Section of the local 'routine' class. This does not affect the way someone defines and uses the variables created within the global section. Variables can be used just as they were in SAP BW 3.X.**

**NOTE:**

If converting a 3.X routine to 04s' and the form (subroutine) has been defined in the global data area, to continue using these subroutines, you can do the following.

1) either convert them to local private methods
**2)** create a subroutine pool (in se38) and execute these subroutines by using the **perform "subroutine" in program "your subroutine pool name".** statement.
3) convert the subroutine logic into a Function Module

**Here you can insert your code for the start routine. In order to abort as you would in a BW 3.X system go ahead and instead of changing the variable abort to a non-zero value, you raise the class exception `cx_rsrout_abort`. For example,**

**`RAISE EXCEPTION TYPE cx_rsrout_abort.`**

**NOTE: To learn more about the RAISE EXCEPTION command reference transaction ABAPHELP**

```
-------------------------*
*       Calculation of source package via sta
rt routine
*---------------------------------------------
-------------------------*
*   <-> source package
*---------------------------------------------
-------------------------*
  METHOD start_routine.
*=== Segments ===

    FIELD-SYMBOLS:
      <SOURCE_FIELDS>    TYPE _ty_s_SC_1.

*$*$ begin of routine -
 insert your code only below this line
 *-*
    ... "insert your code here

*$*$ end of routine -
 insert your code only before this line
   *-*
  ENDMETHOD.                       "start_routin
e
*---------------------------------------------
-------------------------*
*       Method inverse_start_routine
*---------------------------------------------
-------------------------*
*
*       This subroutine needs to be implement
ed only for direct access
*       (for better performance) and for the
Report/Report Interface
*       (drill through).
*       The inverse routine should transform
a projection and
*       a selection for the target to a proje
ction and a selection
*       for the source, respectively.
*       If the implementation remains empty a
ll fields are filled and
*       all values are selected.
*
*---------------------------------------------
-------------------------*
*
*---------------------------------------------
-------------------------*
  METHOD inverse_start_routine.

*$*$ begin of inverse routine -
 insert your code only below this line*-*
    ... "insert your code here
*$*$ end of inverse routine -
 insert your code only before this line *-*

  ENDMETHOD.                       "inverse_star
t_routine
ENDCLASS.                       "routine IMPLEME
NTATION
```

## Individual Routine for Characteristic

```
PROGRAM UPDATE_ROUTINE.
*$*$ begin of global - insert
your declaration only below this
line  *-*
* TABLES: ...
* DATA:   ...
*$*$ end of global - insert your
declaration only before this
line   *-*


FORM compute_key_field
  TABLES   MONITOR STRUCTURE
RSMONITOR "user defined
monitoring
  USING    COMM_STRUCTURE LIKE
/BIC/CS80D_PU_C01
           RECORD_NO LIKE SY-
TABIX
           RECORD_ALL LIKE SY-
TABIX
           SOURCE_SYSTEM LIKE
RSUPDSIMULH-LOGSYS
  CHANGING RESULT LIKE
/BIC/VZD_PU_C01T-D_COUNTRY
           RETURNCODE LIKE SY-
SUBRC
           ABORT LIKE SY-SUBRC.
"set ABORT <> 0 to cancel update
*
*$*$ begin of routine - insert
your code only below this line
*-*
* fill the internal table
"MONITOR", to make monitor
entries

* result value of the routine
  RESULT = .
* if the returncode is not equal
zero, the result will not be
updated
  RETURNCODE = 0.
* if abort is not equal zero,
the update process will be
canceled
  ABORT = 0.
```

```
PROGRAM trans_routine.


*----------------------------------------------------------------------*
*       CLASS routine DEFINITION
*----------------------------------------------------------------------*
*
*----------------------------------------------------------------------*
CLASS routine DEFINITION.
  PUBLIC SECTION.

    TYPES:
      BEGIN OF _ty_s_SC_1,
*      Field: KTOPL Chart of Accts.
        KTOPL           TYPE C LENGTH 4,
*      Field: SAKNR G/L Account No..
        SAKNR           TYPE C LENGTH 10,
*      Field: BUKRS Company Code.
        BUKRS           TYPE C LENGTH 4,
*      Field: GSBER Business Area.
        GSBER           TYPE C LENGTH 4,
      END   OF _ty_s_SC_1.
    TYPES:
      BEGIN OF _ty_s_TG_1,
*      InfoObject: ZPLACTUAL Plan/Actual Indicator.
        /BIC/ZPLACTUAL           TYPE /BIC/OIZPLACTUAL,
      END   OF _ty_s_TG_1.
  PRIVATE SECTION.

    TYPE-POOLS: rsd, rstr.

*$*$ begin of global -
 insert your declaration only below this line
  *-*
    ... "insert your code here
*$*$ end of global -
 insert your declaration only before this line   *-*

    METHODS
      compute_ZPLACTUAL
        IMPORTING
          request                  type rsrequest
          datapackid               type rsdatapid
          SOURCE_FIELDS              type _ty_s_SC_1
        EXPORTING
          RESULT                   type _ty_s_TG_1-/BIC/ZPLACTUAL
          monitor                  type rstr_ty_t_monitor
        RAISING
          cx_rsrout_abort
          cx_rsrout_skip_record.
```

**In both 3.X and 2004s' to set a value for a given characteristic the value of result needs to be set.**

**However as in the start routine above in order to skip a line or abort you need to raise exception. For example:**

**3.X – Abort = 4.**
**04s – raise exception type cx_rsrout_abort.**

**To skip a record in 2004s' insert the following:**

**raise exception type cx_rsrout_skip_record.**

```
    METHODS
      invert_ZPLACTUAL
        IMPORTING
          i_r_selset_outbound          TYPE R
EF TO cl_rsmds_set
          i_th_fields_outbound         TYPE H
ASHED TABLE
          i_r_selset_outbound_complete TYPE R
EF TO cl_rsmds_set
          i_r_universe_inbound         TYPE R
EF TO cl_rsmds_universe
        CHANGING
          c_r_selset_inbound           TYPE R
EF TO cl_rsmds_set
          c_th_fields_inbound          TYPE H
ASHED TABLE
          c_exact                      TYPE r
s_bool.
ENDCLASS.                    "routine DEFINIT
ION

*---------------------------------------------
-------------------------*
*       CLASS routine IMPLEMENTATION
*---------------------------------------------
-------------------------*
*
*---------------------------------------------
-------------------------*
CLASS routine IMPLEMENTATION.

  METHOD compute_ZPLACTUAL.

    DATA:
      MONITOR_REC    TYPE rsmonitor.

*$*$ begin of routine -
 insert your code only below this line
 *-*
... "insert your code here
*--
  fill table "MONITOR" with values of structu
re "MONITOR_REC"
*-   to make monitor entries
... "to cancel the update process
*    raise exception type CX_RSROUT_ABORT.
... "to skip a record"
*    raise exception type CX_RSROUT_SKIP_RECO
RD.

*    result value of the routine
     RESULT = .

*$*$ end of routine -
 insert your code only before this line
   *-*
  ENDMETHOD.                    "compute_ZPLA
CTUAL
*---------------------------------------------
--------------------------*
*       Method invert_ZPLACTUAL
```

```
*--------------------------------------------
--------------------------*
*
*       This subroutine needs to be implement
ed only for direct access
*       (for better performance) and for the
Report/Report Interface
*       (drill through).
*       The inverse routine should transform
a projection and
*       a selection for the target to a proje
ction and a selection
*       for the source, respectively.
*       If the implementation remains empty a
ll fields are filled and
*       all values are selected.
*
*--------------------------------------------
--------------------------*
*
*--------------------------------------------
--------------------------*
  METHOD invert_ZPLACTUAL.

*$*$ begin of inverse routine -
 insert your code only below this line*-*
... "insert your code here
*$*$ end of inverse routine -
 insert your code only before this line *-*

  ENDMETHOD.                     "invert_ZPLAC
TUAL
ENDCLASS.
```

www.sdn.sap.com/irj/sdn/howtoguides