

# Steps to Debug Routines in BI Transformations



## Applies to:

SAP NetWeaver BW 3.x. For more information, visit the [Business Intelligence homepage](#).

## Summary

This article explains the Debugging process for routines in Transformations. It helps us to trace the errors crept in Routines in the debug mode, so that it can be handled effectively.

**Author:** Rudra Pradeep Reddy Neelapu

**Company:** Mahindra Satyam

**Created on:** 31 March 2010

## Author Bio



Working as a SAP Technical consultant with Mahindra Satyam. Skill set includes SAP Business Intelligence, ABAP and Business Objects.

## Table of Contents

Introduction .....	3
Business Scenario .....	3
Procedure .....	3
Generated Program .....	6
Related Content .....	11
Disclaimer and Liability Notice.....	12

## Introduction

This article is mainly intended to provide an idea of how we can handle the debugging process for the routines we code in the transformations. It explains in steps the process to derive at a generated Program which contains our routine code. From here we can carry out with our normal debugging procedure.

## Business Scenario

When we want to know about the errors crept in our routines, we need to debug the code we had written in transformation routines like start and update routines.

## Procedure

Go with the Transformations which need to be debugged.

▼ CSD BSD Outbound Interface_1	ZDM_D_17	Manage
▶ RSDS ZDS_ZISU_DM_T_C_B_LE SBECCLNT300 -> ODSO ZDM_D_17	002NS9TT9TD4NHYRT50F3GGCLPSS3PVO	Change
▶ Data Transfer Processes	ZDM_D_17	Create Data Tra...

Let's have a look at the Start and Update Routine in the Transformations.

Start Routine of the Transformation:

### Transformation Change

```

*$*$ begin of 2nd part global - insert your code only below this line *
... "insert your code here

types: begin of ty_zmdoia,
        UCPREMISE type /BI0/OIUCPREMISE,
        DIVISION type /BI0/OIDIVISION,
        /BIC/ZABLSPERR type /BIC/OIZABLSPERR,
        end of ty_zmdoia,
        tt_zmdoia type standard table of ty_zmdoia.

data: it_zmdoia type tt_zmdoia,
      wa_zmdoia type ty_zmdoia.

Data : ZVAR type /BIC/OIZABLSPERR.

*$*$ end of 2nd part global - insert your code only before this line *
+-----+
+          CLASS routine IMPLEMENTATION
+-----+
+
+-----+
CLASS lcl_transform IMPLEMENTATION.
*$*$ begin of routine - insert your code only below this line *-*
... "insert your code here

    if not SOURCE_PACKAGE is initial.

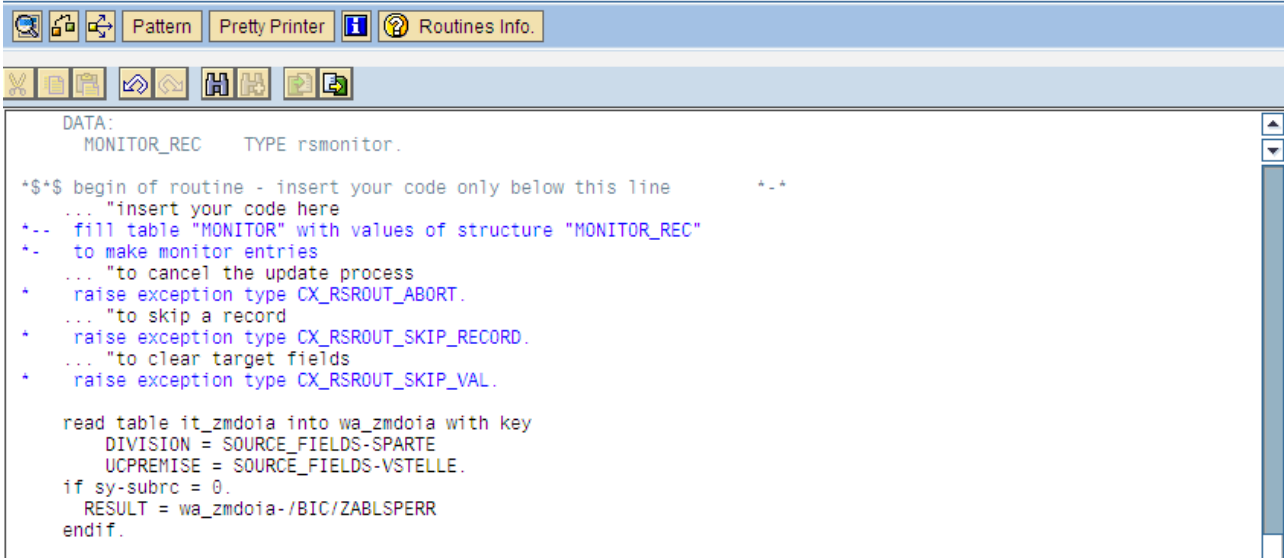
        select UCPREMISE DIVISION /BIC/ZABLSPERR
            from /BIC/AZMDOIA00 into table it_zmdoia
            for all entries in SOURCE_PACKAGE where
                ( ( UCPREMISE = SOURCE_PACKAGE-VSTELLE ) and
                  ( DIVISION = SOURCE_PACKAGE-SPARTE ) ).

        if sy-subrc = 0.
            sort it_zmdoia by UCPREMISE DIVISION.
            delete adjacent duplicates from it_zmdoia
            comparing UCPREMISE DIVISION.
        endif.
    endif.

```

Update Routine:

## Rule Details



```

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.
... "to clear target fields
* raise exception type CX_RSROUT_SKIP_VAL.

read table it_zmdoia into wa_zmdoia with key
  DIVISION = SOURCE_FIELDS-SPARTE
  UCPREMISE = SOURCE_FIELDS-VSTELLE.
if sy-subrc = 0.
  RESULT = wa_zmdoia-/BIC/ZABLSPERR
endif.

```

To debug these routines we need to go to the generated program of the Transformations in two ways:

We can get the Program ID for the **Transformation ID (4H8JA4TG3JNQO0VTU3HK01MH7)** from table **RSTRAN**.

Program Edit Goto Settings System Help

Number of Entries

Transformation ID	002NS9TT9TD4NHVRT5	to		→
Version	A	to		→
Object Status		to		→
Content release		to		→
Cont.time stamp		to		→
Person Respons.		to		→
BW Application		to		→
active		to		→
Last changed by		to		→
UTC Time Stamp in Short Form (		to		→
Object Type		to		→
Subtype TLOGO		to		→
Object Name		to		→
Object Type		to		→
Subtype TLOGO		to		→
Object Name		to		→
ID		to		→
ID		to		→
ID		to		→
ID		to		→
Program ID		to		→

Table Entry Edit Goto Settings System Help

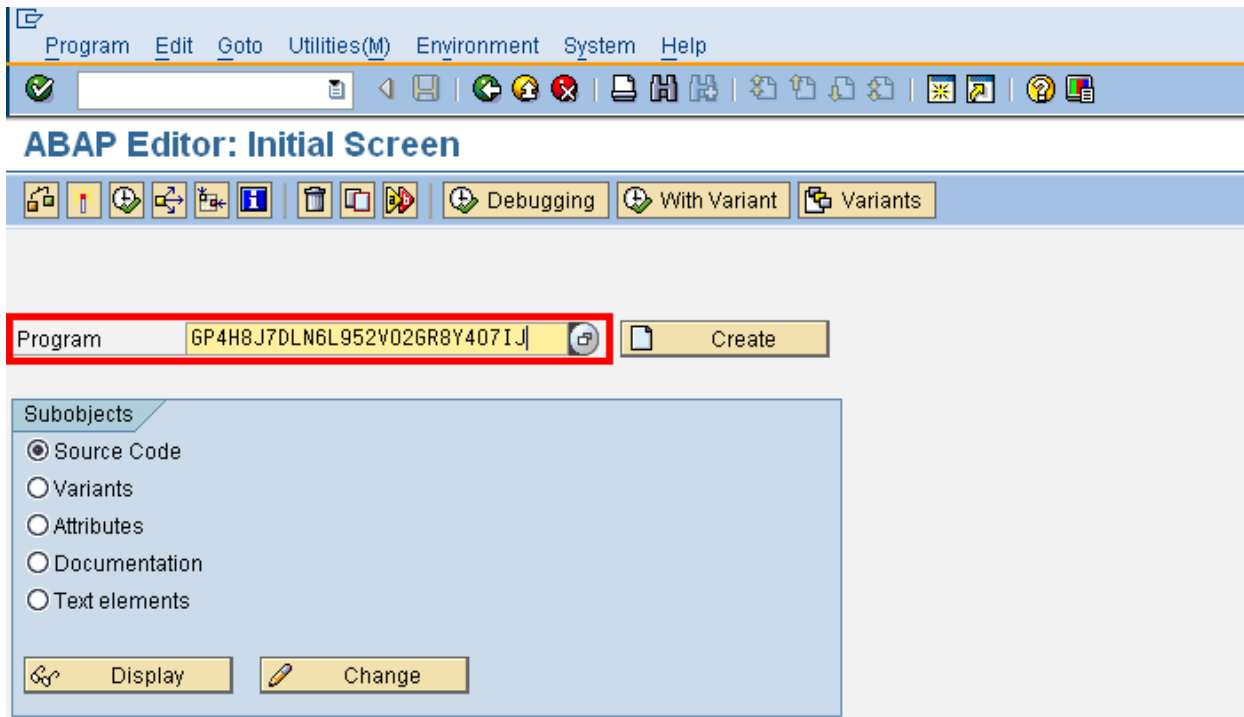
Data Browser: Table RSTRAN Select Entries 1

Transformation ID	V	Obj. Type	Object Name	Object Name	ID	GLBCODE	ProgID
002NS9TT9TD4NHVRT50F3GGCLPSS3PV0	A	RSDS	ZDS_ZISU_DM_T_C_B_LE	SBECLNT300	ZDM_D_17	4H8JA4LRKL215ECD09F7PZNR	4H8J7DLN6L952VO2GR8Y407IJ

On Appending the ProgID (4H8J7DLN6L952VO2GR8Y407IJ) with GP we get the report program for the Transformations.

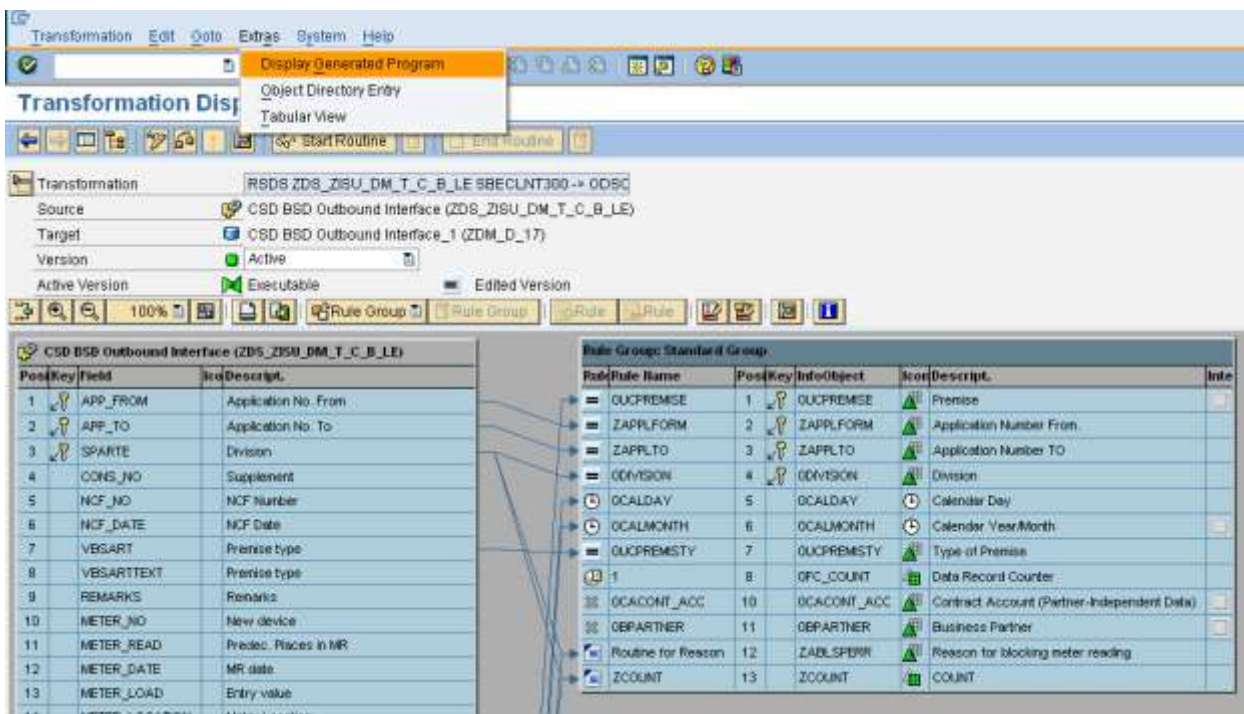
Go to SE38 and give the Program Name (GP4H8J7DLN6L952VO2GR8Y407IJ).

## Generated Program



On going with display we will be to the ABAP editor of the Report Program. The other way of getting to this program is from the Transformations itself.

Go to Extras → Display Generated Program. It directly takes us to the ABAP editor of the Report Program.



Below is the ABAP editor screen of the Report Program:

```

*&
*& Include          RSTRAN_MASTER_TMPL
*&
*&
*& Template RSTRAN_MASTER_TMPL for transformation runtime
*&-----*
*
*
* Generated Runtime for Transformations
*
* Template.....: RSTRAN_MASTER_TMPL
* TranID.....: 0156W18KFT4XUXLSW162BQUUYCBM7XH2   Version: 0
* Source.....: RSDS 2LIS_18_I0CAUSE                SBCLNT300
* Target.....: CUBE ZCS_C_06
*
* Author.....: SATHISHK
* Date.....: 17.03.2010 15:08:26
*
* Do not change this source !
*
*-----*
program RSTRAN_MASTER_TMPL.
*-----*
*
* CLASS lc1_transform DEFINITION

```

Set a break point at the code you want to debug, i.e. at your selections at start routine and at your update routine.

```

DATA:
  MONITOR_REC      TYPE rstmonitor.
  ... "insert your code here

if not SOURCE_PACKAGE is initial.

  select UCPREMISE DIVISION /BIC/ZABLSPERR
    from /BIC/AZMDOI00 into table it_zmdoia.
  if sy-subrc = 0.
    sort it_zmdoia by UCPREMISE DIVISION.
    delete adjacent duplicates from it_zmdoia
      comparing UCPREMISE DIVISION.
  endif.
endif.

*-- fill table "MONITOR" with values of structure "MONITOR_REC"
*- to make monitor entries
... "to cancel the update process
* raise exception type CX_RSROUT_ABORT.

ENDMETHOD.                "start_routine
METHOD inverse_start_routine.

*$$$ begin of inverse routine - insert your code only below this line*-
* raise exception type CX_RSROUT_SKIP_VAL.

  read table it_zmdoia into wa_zmdoia with key
    DIVISION = SOURCE_FIELDS-SPARTE
    UCPREMISE = SOURCE_FIELDS-VSTELLE .

  if sy-subrc = 0.
    RESULT = wa_zmdoia-/BIC/ZABLSPERR.
  endif.

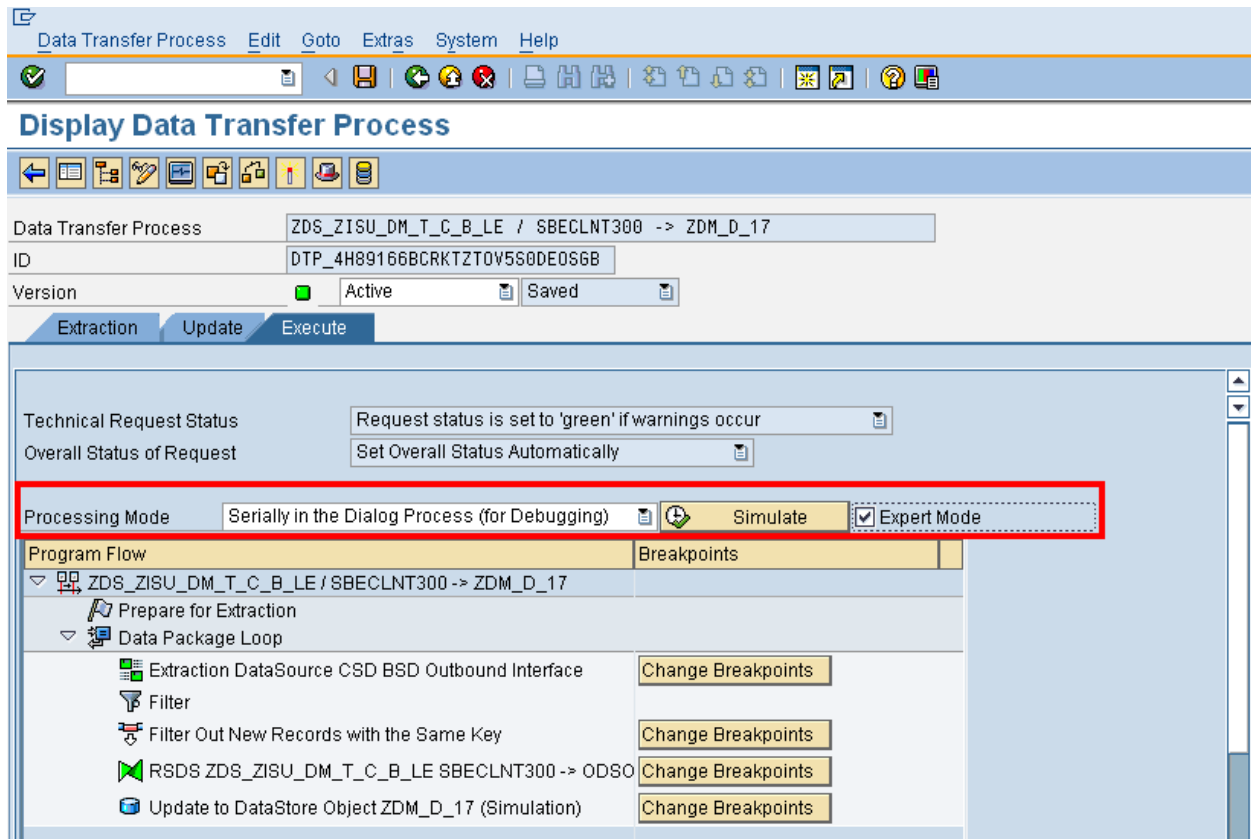
ENDMETHOD.                "compute_12_1
METHOD invert_12_1.

```

We can exit the screen and can debug it by running the DTP in Debugging Mode.

▼ CSD BSD Outbound Interface_1	ZDM_D_17	Manage
▶ RSDS ZDS_ZISU_DM_T_C_B_LE SBECINT300 -> ODSO ZDM_D_17	002NS9TT9TD4NHYRT50F3GGCLP8S3PVO	Change
▼ Data Transfer Processes	ZDM_D_17	Create Data Tra...
▶ ZDS_ZISU_DM_T_C_B_LE / SBECINT300 -> ZDM_D_17	DTP_4H89166BCRKKTZTOV5S0DE0SGB	Change





Select the Processing Mode Serially in the Dialog process (for Debugging) and go with simulate with Expert Mode. If you set this indicator, you can debug a DTP request in expert mode. The system does not execute the simulation request directly; you can edit the simulation request interactively before it is processed.

You can change the following properties:

- Selections
- Setting for temporary storage
- Breakpoints

We can use expert mode in the following cases:

- If the amount of data in the source is very large and the runtime of a simulation request for the entire selection would be considerable
- If you want to generate the temporary storage during request processing but do not want to change the DTP settings.
- This is useful in productive systems where you cannot change DTPs.
- If you want to test a particular record

On going with Simulate we are with the ABAP Debugger Session. Now we can go with normal debugging process.

The screenshot displays the SAP ABAP Debugger interface. The main window shows the source code of the routine `START_ROUTINE`. A red box highlights the following code segment:

```

1206  if not SOURCE_PACKAGE is initial.
1207
1208  select UCPRENISE DIVISION /BIC/ZARLSPERR
1209         from /BIC/AZND01A08 into table it_zmdoia.
1210  if sy-subrc = 0.
1211  sort it_zmdoia by UCPRENISE DIVISION.
1212  delete adjacent duplicates from it_zmdoia
1213  comparing UCPRENISE DIVISION.
1214  endif.
1215  endif.

```

To the right, the 'Variables' window shows the current values of the variables `SOURCE_PACKAGE` and `IT_ZMDOIA`, also highlighted with a red box:

Variable	Type	Value
SOURCE_PACKAGE	Standard-Table [218x58 (1868)]	
IT_ZMDOIA	Standard-Table [11202x3 (20)]	

Here we can find the entries in the `SOURCE_PACKAGE` and the entries fetched into the internal table `IT_ZMDOIA` in the start routine.

## Related Content

Taken most of the inputs from **sap help** in preparing this Article.

<http://forums.sdn.sap.com/click.jspa?searchID=41376436&messageID=8664110>

<http://forums.sdn.sap.com/click.jspa?searchID=41376436&messageID=8646524>

For more information, visit the [Business Intelligence homepage](#).

## Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.