How To...
Create Generic DataSources which use the Delta Queue

Version 1.00 – February 2006

Applicable Releases:
SAP NetWeaver ‘04
(SAP BW 3.x)
1 Business Scenario

You have been tasked to bring purchase requisitions into your data warehouse and you have not been able to identify a standard content dataSource which will accomplish this task for you.

Using the generic datasources available to you in SAP BW is your preferred solution however you have not been able to identify a date field which accurately captures delta records.

You need to deliver a solution to appropriately capture deltas within your purchasing system for purchase requisitions.

2 Introduction

Within SAP R/3 and mySAP ERP systems there are Business Transaction Events or BTE’s. BTE’s are essentially events that take place at the time of a business transaction such as a change to a purchase order or for our particular scenario a purchase requisition. Keep in mind there are a great deal of different types of business transaction events in all areas of SAP R/3, each event can indeed enable a delta capture mechanism. Please consult transaction FINF to navigate through the existing BTE’s in your SAP R/3 system and educate yourself further.

These events can be leveraged to capture delta changes and write these changes to the delta queue directly within transactional systems. BTE’s act as a “delta hook”, this enables the use of standard delta queue functionality within your data warehouse as long as your datasource is configured appropriately.

Certain delta processes within datasources support the use of the delta queue whereas others do not. In order to implement this solution a dataSource will have to be configured with a delta process such as ABR, or AIM that inherently support the use of the delta queue as a delta capture mechanism.

3 Step by Step Solution

The task of creating a dataSource which retrieves purchase requisition line item information from a transactional (SAP R/3) system moving this information into a data warehouse (SAP BW) has been assigned to us.
3.1 Find a BTE to use for delta capture

1. Navigate to transaction FINF and identify a business transaction event, BTE which captures changes to the table(s) needed.

2. Identify the BTE which captures the necessary information.
3. Create a function module to take care of the delta write functionality being built into our dataSource.

A sample function module interface can be found on the screen we saw in the last step (it is circled in red). By clicking on the Sample Function Module button the Function Module Builder transaction will then start up where the Sample Function Module can be copied as can be seen here on the right.

Name and save this function module because it will be used again in step 3.3.

Note: Within Business Transaction Events two types of internal tables are passed for each database table, an X table and a Y table. The Y table contains the original values of the records modified, whereas the X table contains the new records with the modified values.

3.2 Create a View

1. Navigate to transaction SE11 within the SAP R/3 system. Create a new view which is built on top of the EBKN table.

NOTE: Here a view is being created in order to be used within the Generic DataSource. If it is preferable to the user to create a FM or InfoSet for the Generic DataSource, this can be done as well.
2. Next, run the view and make sure that all of the necessary data required is available from this view.

3.3 Create the datasource

1. Navigate to transaction RSO2.

Fill in the appropriate textual information and the name of the database view just created. Hit the save button.

**NOTE:** The DataSource must begin with a 'Z_'.
2. Choose the fields which will be used for selection criteria and then save the DataSource.

3. Now, it is necessary to modify the delta process type of the newly created dataSource to one that uses the delta queue. To do this there is a utility program created (see appendix), Z_CHANGE_DELTA_PROCESS.
   a. This program needs to be installed, to do this navigate to transaction se38 and type in the program name.
   b. Hit the create button.
   c. Copy and paste the code from the appendix into the ABAP Editor and activate it.
4. To modify the delta process type, run this program. You will see two parameters that you need to fill in to run the program.
   a. The DataSource name
   b. The Delta Process type

   The Delta Process type you enter needs to support the delta queue. For this example we will use 'AIM', After Images using the Delta Queue.

5. Once the necessary parameters have been entered execute the program. This will update the dataSource appropriately.

3.4 Create Product and Link BTE & Function Module

1. Navigate to transaction FIBF and find the menu path to create a new Product.
2. From here a customer product is created, this is necessary in order to invoke a function module at the time the business transaction event occurs. It is a good idea to maintain some sort of naming convention for both the products created as well as their associated Function Modules.

3. A link needs to be created between our BTE, the Function Module created and the Product, Z_DELTA created in the last step. To do this find the P/S Function Module of a customer option using the following path Settings->P/S Function Modules-> ... of a customer.

4. Create a new entry specifying the BTE number, the product created as well as the function module shell we created in step 3.1. Save this setting.
### 3.5 Finish Delta Hook

1. Navigate to transaction se37 create a new Function module and give it the name **Z_WRITE_TO_QUEUE**.

2. Copy and paste the code from the Appendix 4.2 into the Function Module Editor, then save and activate the Function Module.

   **NOTE:** This function module will be used later to write to the delta queue. The standard extraction user exit is also called within this Function Module; all enhancements created via the standard mechanism are supported.
3. Now navigate to the other Function Module we created in step 3.1 and fill it with the necessary code to write to the delta queue.
   
   a. Navigate to transaction se37
   b. Place the name of the function module in the input field and hit change.
   c. Insert the code from the Appendix 4.3.
   d. Insert the name of your dataSource created
      
      i. `l_datasource = 'Z_BTE_DELTA'.`  
   e. The last step is to specify the X table passed from the BTE interface within the call to the Function Module `Z_WRITE_TO_QUEUE`, in our example we want to capture Purchase Requisition Account Assignment changes.

   Now save and activate the Function Module.

3.6 Test New Functionality

1. Login to the BW system, once in the system perform the following steps
   
   a. Navigate to transaction RSA1
   b. Under the Source System tab replicate the dataSources of the system in which you implemented this How-To.
   c. Find the dataSource you created in the DataSource Overview Pane.
2. Navigate to the InfoSources tab and create an InfoSource that matches the Extraction Structure of your dataSource. Save and activate.
   
   a. Go back to the Source System tab and find your DataSource again
   
   b. Click on the red sign and assign the infoSource created to your DataSource.

3. Now that the infoSource is created and the DataSource linked, create an infoPackage for the delta initialization. For the purposes of this How-To we will load into the PSA only. Now execute the delta init.

4. Verify the data within the PSA
5. Check the delta queue in your source system via transaction RSA7 and make sure a delta queue entry was created successfully.

6. To verify that the delta hook built is working we now need to modify an existing purchase requisition and make a change to the Account Assignment information; this will then invoke the delta hook.
7. Verify in the delta queue the successful write of the modified record.

8. In the BW system create another infoPackage and set it up for a delta update.
9. Kick off the infoPackage and validate the information within the PSA.
4 Appendix A: Example ABAP Code

4.1 Z_CHANGE_DELTA_PROCESS

*&---------------------------------------------------------------------*
*& Report Z_CHANGE_DELTA_PROCESS                                      *
*&                                                                     *
*&---------------------------------------------------------------------*
*&                                                                     *
*&                                                                     *
*&---------------------------------------------------------------------*
report  z_change_delta_process                  .
*P_DATAS DataSource
*P_DELTAP Delta Process for DataSource
parameters:
  p_datas  type roosource-oltpsource,
  p_deltap type roosource-delta.

tables:
  roosource.

data:
  ls_roosource type roosource.

if p_datas(2) ne 'Z_'.
  message 'The DataSource needs to begin with ''Z_''.' type 'E'.
endif.

select single * from  roosource into ls_roosource
  where  oltpsource  = p_datas
      and    objvers     = 'A'.

if sy-subrc eq 0.
  ls_roosource-delta = p_deltap.
  update roosource from ls_roosource.
  message 'The DataSource has been updated successfully.' type 'I'.
else.
  message 'The DataSource entered is not valid, try again.' type 'E'.
endif.

4.2 Z_WRITE_TO_QUEUE

FUNCTION z_write_to_queue.

*"----------------------------------------------------------------------
*"Local interface:
*"IMPORTING
*"  REFERENCE(I_DATASOURCE) TYPE  ROOSOURCE-OLTPSOURCE
*"  TABLES
*"  I_T_DATA OPTIONAL
*"----------------------------------------------------------------------

* Developed by: Scott Cairncross

TYPE-POOLS:
  sbiwa.

DATA:
TABLES
  t_fields = t_fields
  c_t_data = e_t_data
EXCEPTIONS
  rsap_customer_exit_error = 1
  OTHERS = 2.
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.

CALL FUNCTION 'RSC1_TRFC_QUEUE_WRITE'
  EXPORTING
    i_isource = i_datasource
i_no_flush = 'X'
IMPORTING
e_subrc = l_subrc
TABLES
i_t_data = <e_t_data>
EXCEPTIONS
name_too_long = 1
OTHERS = 2.
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
  WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.
ENDIF.
ENDFUNCTION.

*---------------------------------------------------------------------*
*       FORM abs_type_to_rel_type                                     *
*---------------------------------------------------------------------*
*       The purpose of this subroutine is to convert an absolute type *
*       name into a relative type name.                               *
*                                                                     *
*---------------------------------------------------------------------*
*  -->  TYPE_NAME                                                     *
*---------------------------------------------------------------------*
form abs_type_to_rel_type changing type_name.
data junk(100) type c.
  split type_name at '\TYPE=' into junk type_name.
endform.

4.3  ZSC_BTE_DELTA_CAPTURE_01000710

function zsc_bte_delta_capture_01000710.
*"----------------------------------------------------------------------
*"Local interface:
*" TABLES
*"  T_XEBAN STRUCTURE  EBAN OPTIONAL
*"  T_XEBKN STRUCTURE  EBKN OPTIONAL
*"  T_YEBAN STRUCTURE  EBAN OPTIONAL
*"  T_YEBKN STRUCTURE  EBKN OPTIONAL
*"----------------------------------------------------------------------
data:
  l_datasource type roosource-oltpsource.
l_datasource = 'Z_BTE_DELTA'.
if lines( t_xebkn[] ) gt 0.
  call function 'Z_WRITE_TO_QUEUE'
    exporting
      i_datasource = l_datasource
      tables
        i_t_data = t_xebkn.
  endif.
endfunction.

endfunction.