Three Ways to Dynamically Update “Data Transfer Process” Filter Values

Applies to:
SAP BI 7.0. For more information, visit the Business Intelligence homepage.

Summary
The objective of this article is a how-to document for getting a clear understanding on how to supply dynamic filter values for DTP.

Author: Vasu Sattenapalli
Company: Intellepeople, Inc.
Created on: 1 January 2010

Author Bio
Vasu Sattenapalli is a SAP BI Solution Architect has been working on data warehousing and business intelligence implementations since 1999. He had extensively worked on business blue print, requirements analysis, architecture, designing, development including configuration and ABAP coding, resource and project management of world class SAP BI implementation and production support projects.
# Table of Contents

**Introduction** ...................................................................................................................................................... 3

**Using DTP Routine** ............................................................................................................................................. 3
  Business Scenario: .................................................................................................................................................. 3
  Caution: ................................................................................................................................................................. 7
  End result: ............................................................................................................................................................. 7

**Using OLAP Variables** ......................................................................................................................................... 8
  Business Scenario .................................................................................................................................................. 8
  Step 1: Create OLAP Variable ............................................................................................................................... 8
  Step 2: Enhance User-exit include “zxrsru01” ........................................................................................................ 11
  Step 3: Use the Variable in DTP Filter .................................................................................................................. 12
  End result: ............................................................................................................................................................. 14

**Using Custom ABAP Program to Update DTP Filters** .......................................................................................... 14
  Business Scenario .................................................................................................................................................. 14
  Create Function Module to Update DTP Filter ........................................................................................................ 15
  ABAP Program to Determine Filter Values ............................................................................................................. 20
  End result: ............................................................................................................................................................. 21

**Related Content** .................................................................................................................................................... 22

**Disclaimer and Liability Notice** .......................................................................................................................... 23
Introduction

Data Transfer Process, AKA DTPs are introduced in SAP BI 7.0, which are used for transferring the data from PSA-to-Infoprovider, Infoprovider-to-Infoprovider or Infoprovider-to-Open Hub Destination within the BI instance. Similar to infopackages, it is possible to define the data selection criteria for DTPs also.

Apart from the static hardcoded filter values, following are the three provisions to supply the filter values dynamically.

- DTP Routine
- OLAP Variables
- Custom ABAP Program

In this article I will take a business scenario to explain each of the above approaches.

Using DTP Routine

Business Scenario:

The business requirement is to have a monthly sales snapshot for evaluating Open Orders. Architecturally this has been enabled by using daily sales DataStore Object (DSO) that provides the monthly snapshot of Sales Data to a Monthly Sales InfoCube (Cube) used in reporting. The goal is to enable an automatic process that performs the Monthly snapshot load from the DSO to the Cube for the previous month’s activity on the 1st day of every fiscal period.

In the example below the DSO YSLSORD ‘Sales order Demo’ is used along with the cube YSLSORD ‘Sales Order Demo’. The key reporting metrics for Open Order evaluation are “Open Order Quantity” and “Open Order Amount”. Open order quantity and amounts are available in YSLSORD (Sales Orders Demo) Data Store Object. Snapshot data needs to be loaded into infocube YOPNSNAP (Open Order Snapshot Demo). This extraction should happen on first day of every fiscal period. Dataflow is as show in Figure 1.

![Figure 1: Data Flow Diagram for DTP Routine Demo](image-url)
DTP needs to select the list of previous fiscal period’s sales orders and load into Infocube YOPNSNAP. Since the previous fiscal period value has to be derived during runtime, it can’t be persisted in the DTP filter value. For this we can to use DTP routine.

For example if the current fiscal period is 011/2008, when DTP is executed it should dynamically change the DTP fiscal period filter value to 010/2008.

In order for creating a dynamic filter on the field “FISCPER”, go to DTP in change mode, Click on “Filter” button. This will display all the list of fields from source of the DTP. Locate the “Fiscal Year/Period” field, click on the right most blue rectangle icon is for DTP routine as show in Figure 2. Give a name to routine as “Get Previous Fiscal Period”, Click on editor icon.

Figure 2: Create DTP Filter on Fiscal Year/Period field

This will take to the ABAP editor with system generated ABAP code as show in Figure 3.
Three Ways to Dynamically Update “Data Transfer Process” Filter Values

Figure 3: Code that automatically generated
Modify the above generated as shown in Figure 4.

Click on save button on the editor, this takes back to the DTP. Execute the DTP, click on DTP monitor, Go to DTP monitor header tab as show in Figure 5. You can see in the DTP monitor selection field is populated with the previous fiscal period.
Caution:

Applies to SAP BI 7.0 support pack lower than 15, be aware of potential issue with DTP routine documented in OSS note 1057820 (70SP15: Filter with routine deletes selections).

End result:

On the 1st day of every month the DTP loads a snapshot of the previous month data via a scheduled process chain load. No manual intervention is required to determine or update parameters for the current month or previous month for the load process. Also this is all accomplished using a single DTP!
Using OLAP Variables

Business Scenario

Weekly Sales data from SAP BI is needed to provide in the form of flat file to non BI systems for planning and forecasting purposes. The goal is to automate the process of running the weekly extract on the 1st day of every week for the previous week’s Sales.

In this example I need to extract sales data from sales DSO YSLSORD and create a flat file. File should contain previous calendar week’s data and extract need to run first day of every calendar week.

In SAP BI 7.0, it is recommended to use open hub destination to create flat file out of data from SAP BI infoproviders. For example, if the current calendar week is 48/2008, when DTP is executed it should dynamically change the DTP calendar week filter to 47/2008.

This method is a three step process.

1. Create OLAP Variable for 0CALWEEK
2. Enhance User-exit include “zxrsru01”
3. Use the Variable in DTP Filter

Step 1: Create OLAP Variable

Create OLAP characteristic variable on 0CALWEEK variable YVAR_PRCALWK. For information about OLAP characteristic variable, please see SAP documentation using the following URL.


In BEX Query designer, locate time characteristic 0CALWEEK, right click and choose option “New Variable” from context menu as show in Figure 7.
Change technical name to YVAR_PRCALWK, change processing by option to "Customer exit" as shown in figure 8. Click on save.

Figure 7: Create OLAP Variable on 0CALWEEK

Figure 8: OLAP Variable Details
**Note:** By default when you create a characteristic variable the option “Variable is Ready for Input” is selected. In order to use the OLAP variable in DTP filter this option should be unchecked as shown in figure 9.

![Change Variable](image)

**Figure 9:** OLAP Variable Details
Step 2: Enhance User-exit include “zxrsru01”

Update the userexit ZXRSU01 with logic to populate YVAR_PRCALWK variable with prior calendar week. In for this logic to be invoked during the DTP filter, code needs to be written in i_step = 0.

Write the ABAP code as show in the figure 10.

![ABAP Code in userexit ZXRSRU01](image)

Save and activate the userexit.
Step 3: Use the Variable in DTP Filter

In order for creating a dynamic filter on the field 0CALWEEK, go to DTP in change mode, Click on “Filter” button. This will display all the list of fields from source of the DTP. Locate the field “Calendar Year/Period”, click on the yellow rectangle icon is for DTP routine as show in Figure 11

![Filter Dialog Box](image)

**Figure 11**: Create DTP Filter using OLAP Variable

Click on continue, this will open the dialog box where we can input OLAP Variable name as show in figure 12.
Figure 12: Select OLAP Variable

Click on continue on “Use Variable of OLAP processor” dialog box and filter dialog box, this takes back to the DTP. Execute the DTP, click on DTP monitor, Go to DTP monitor header tab as show in Figure 12. You can see in the DTP monitor selection field is populated with the previous calendar week.
Figure 12: DTP monitor for OLAP Variable option

Note: In order for OLAP variable option in DTP to work, even though fiscal year variant is not needed for OLAP variable to be derived, it is mandatory to provide fiscal year variant

End result:

On the 1st day of every week DTP loads a snapshot of the previous calendar week data via a scheduled process chain load. No manual intervention is required to determine or update parameters for the current calendar week or previous calendar week for the load process.

Using Custom ABAP Program to Update DTP Filters

Business Scenario

The same scenario used in “OLAP Variables” option is used and fulfill the requirement using Custom ABAP program approach

With this approach, we use an external custom ABAP program to update DTP filter values. This approach can be used if there is a complicated logic involved to determine the filter values.

This is a three step approach.

- Create Generic DTP Filter Update Program (Re-usable)
- ABAP Program to determine Filter Values (Specific to Business Scenario)
Create Function Module to Update DTP Filter

This is a generic program that accepts DTP name, DTP filter field name, DTP filter value as input. By running this program this will update and activate the given DTP by changing the DTP filter values as per the input.

Dataflow diagram for Custom ABAP Program Approach
Following is the source code for this program.

```
FUNCTION Y_CHANGE_DTP_FILTERS_2.
  "---------------------------------------------------------------------
  & Author: Vasu Sattenapalli
  & Functionality: This is a generic function module that accepts DTP
  & name, DTP filter fields and filter values as input and updates the
  & corresponding DTP with information provided as INPUT.
  &---------------------------------------------------------------------
  & Update History:
  &---------------------------------------------------------------------
  & Person Date Desc
  "---------------------------------------------------------------------
  & Highlevel steps in program logic
  & Step 1: get existing filter values (selection table values)
  & Step 2: Change the status of DTP to INACTIVE, This will create M version of DTP
  & Step 3: Update selection table of DTP with internal table from step 2
  & Step 4: Save and Activate the DTP, this will save the changes to M version and Activates.
  "---------------------------------------------------------------------
  "**Local Interface:
  "** IMPORTING
  "**  REFERENCE(P_DTP) TYPE RSBKDTP-DTP
  "**  REFERENCE(P_FLVL1) TYPE STRING
  "**  REFERENCE(P_FLVL2) TYPE STRING OPTIONAL
  "**  REFERENCE(P_FLVL3) TYPE STRING OPTIONAL
  "**  REFERENCE(P_FLVL4) TYPE STRING OPTIONAL
  "**  REFERENCE(P_FLVL5) TYPE STRING OPTIONAL
  "**  REFERENCE(P_OPTION1) TYPE RSOPTION DEFAULT 'EQ'
  "**  REFERENCE(P_OPTION2) TYPE RSOPTION DEFAULT 'EQ'
  "**  REFERENCE(P_OPTION3) TYPE RSOPTION DEFAULT 'EQ'
  "**  REFERENCE(P_OPTION4) TYPE RSOPTION DEFAULT 'EQ'
  "**  REFERENCE(P_OPTION5) TYPE RSOPTION DEFAULT 'EQ'
  "**  REFERENCE(P_FLVL1_TO) TYPE STRING OPTIONAL
  "**  REFERENCE(P_FLVL2_TO) TYPE STRING OPTIONAL
  "**  REFERENCE(P_FLVL3_TO) TYPE STRING OPTIONAL
  "**  REFERENCE(P_FLVL4_TO) TYPE STRING OPTIONAL
  "**  REFERENCE(P_FLVL5_TO) TYPE STRING OPTIONAL
  "**---------------------------------------------------------------------
  DATA: lcl_dtp TYPE REF TO cl_rsbk_dtp,
        lcl_request TYPE REF TO cl_rsbk_request,
        lcl_filter TYPE REF TO cl_rsbc_filter,
        w_loc_dtp TYPE rsbdtpnm.
  DATA:
    s_sign TYPE rssign VALUE 'I',
    s_option TYPE rsoption VALUE 'EQ'.
```
w_loc_dtp = p_dtp.
lcl_dtp = cl_rsbk_dtp=>factory( w_loc_dtp ).

DATA:
g_t_dtprule TYPE mch_t_sourcecode,
g_s_varseltab TYPE mch_var_select,
g_t_varseltab TYPE mch_t_var_select,
g_s_selaltab TYPE rsbk_s_select,
g_t_selaltab TYPE TABLE OF rsbk_s_select,
g_s_selfields TYPE mch_s_field,
g_t_selfields TYPE mch_t_field,
g_r_filter TYPE REF TO cl_rsbc_filter.

lcl_filter = lcl_dtp->get_obj_ref_filter( ).

*Step 1: get existing filter values (selection table values)
CALL METHOD lcl_filter->get_all
IMPORTING
  e_t_varseltab = g_t_varseltab
  e_t_selaltab = g_t_selaltab
  e_t_dtprule = g_t_dtprule
  e_t_selfields = g_t_selfields.

*Collect the filter value changes into internal table
CLEAR:
g_t_selaltab, g_s_selaltab.

IF p_flob1 IS NOT INITIAL AND p_flvl1 IS NOT INITIAL.
g_s_selaltab-field = p_flob1. "PLANT".
g_s_selaltab-sign = s_sign.
g_s_selaltab-option = P_OPTION1.
g_s_selaltab-low = p_flvl1.
g_s_selaltab-HIGH = p_flvl1_to.
APPEND g_s_selaltab TO g_t_selaltab.
CLEAR g_s_selaltab.
ENDIF.

IF p_flob2 IS NOT INITIAL AND p_flvl2 IS NOT INITIAL.
g_s_selaltab-field = p_flob2. "PLANT".
g_s_selaltab-sign = s_sign.
g_s_selaltab-option = P_OPTION2.
g_s_selaltab-low = p_flvl2.
g_s_selaltab-HIGH = p_flvl2_to.
APPEND g_s_selaltab TO g_t_selaltab.
CLEAR g_s_selaltab.
ENDIF.

IF p_flob3 IS NOT INITIAL AND p_flvl3 IS NOT INITIAL.
g_s_selaltab-field = p_flob3. "PLANT".
g_s_selaltab-sign = s_sign.
g_s_selaltab-option = P_OPTION3.
g_s_selaltab-low = p_flvl3.
g_s_seltab-HIGH = p_flvl3_to.
APPEND g_s_seltab TO g_t_seltab.
CLEAR g_s_seltab.
ENDIF.

IF p_flob4 IS NOT INITIAL AND p_flvl4 IS NOT INITIAL.
g_s_seltab-field = p_flob4. "PLANT".
g_s_seltab-sign = s_sign.
g_s_seltab-option = P_OPTION4.
g_s_seltab-low = p_flvl4.
g_s_seltab-HIGH = p_flvl4_to.
APPEND g_s_seltab TO g_t_seltab.
CLEAR g_s_seltab.
ENDIF.

IF p_flob5 IS NOT INITIAL AND p_flvl5 IS NOT INITIAL.
g_s_seltab-field = p_flob5. "PLANT".
g_s_seltab-sign = s_sign.
g_s_seltab-option = P_OPTION5.
g_s_seltab-low = p_flvl5.
g_s_seltab-HIGH = p_flvl5_to.
APPEND g_s_seltab TO g_t_seltab.
CLEAR g_s_seltab.
ENDIF.

*Step 2: Change the status of DTP to INACTIVE, This will create M version of DTP

TYPES:      rs_objstat           TYPE rs_char3.
CONSTANTS:  BEGIN OF rs_c_objstat,
            active     TYPE rs_objstat VALUE 'ACT',
            inactive   TYPE rs_objstat VALUE 'INA',
            off        TYPE rs_objstat VALUE 'OFF',
            productive TYPE rs_objstat VALUE 'PRO',
            all        TYPE rs_objstat VALUE '%',
            END OF rs_c_objstat.

TRY.

CALL METHOD lcl_dtp->if_rsbk_dtp_maintain~set_objstat
EXPORTING
  i_objstat = rs_c_objstat-inactive.

ENDTRY.

CATCH cx_rs_failed.

*Step 3: Update selection table of DTP with internal table from step 2

CALL METHOD lcl_filter->set_all
EXPORTING
  i_t_seltab    = g_t_seltab
  i_t_selfields = g_t_selfields
  i_t_dtprule   = g_t_dtprule
  i_t_varseltab = g_t_varseltab.

* Step 4: Save and Activate the DTP, this will save the changes to M version and Activates.
lcl_dtp->save( ).
lcl_dtp->activate( ).
COMMIT WORK.

DATA: output TYPE string.

LOOP AT g_t_seltab INTO g_s_seltab.
  CONCATENATE 'DTP "" p_dtp "" INTO output SEPARATED BY space.
  CONCATENATE output 'Filter Value for the field' g_s_seltab-field INTO output SEPARATED BY space.
  CONCATENATE output 'is updated to' g_s_seltab-low INTO output SEPARATED BY space.
  WRITE output.
ENDLOOP.

ENDFUNCTION.

************************************************************************************
****************************
******
ABAP Program to Determine Filter Values

This program will be specific for each DTP that requires update of filter values dynamically.

Following is the source code to determine “Previous Fiscal Week” and to update DTP with that value. This program invokes the function module Y_CHANGE_DTP_FILTERS, which accepts DTP name, DTP filter field and filter field as input and updates DTP values with input and activates the DTP.

Source code for this program is as show in the Figure 16.

```
REPORT update_dtp_fiscal_week.
PARAMETERS: i_dtp TYPE sbbdtp-dtp OBLIGATORY,
  i_fascid (40) TYPE c DEFAULT '/BIC/YFISCH',
  i_fiscwk TYPE soal-week, "To overwrite.
DATA: v_today TYPE sy-datum,
  v_lastday TYPE sy-datum,
  v_fiscweek TYPE soal-week,
  v_prior_wk TYPE soal-week,
  v_year TYPE t0000b-secy,
  v_weekly TYPE alweek.

* IF fiscal week overwrite is not provided
  * Find the current day's fiscal week.
  IF i_fiscwk IS INITIAL.
    v_today = sy-datum.
    " Get the current Fiscal Week
    CALL FUNCTION 'Y_GET_CALFISC_WEEK_FROM_DATE'
      EXPORTING
        v_date = v_today
        v_fiscal = 'T1'
    IMPORTING
        v_fisc_week = v_fiscweek.

* Get begining date of current fiscal week.
    CALL FUNCTION 'Y_GET_FISCAL_WEEK_BEG_END_DATE'
      EXPORTING
        v_fiscwk = v_fiscweek
        v_fiscal = 'Y1'
    IMPORTING
        v_begdate = v_lastday.
    v_lastday = v_lastday - 1.
    CALL FUNCTION 'Y_GET_CALFISC_WEEK_FROM_DATE'
      EXPORTING
        v_date = v_lastday
        v_fiscal = 'T1'
    IMPORTING
        v_fisc_week = v_prior_wk.
    v_fiscweek = v_fiscweek.
    ELSE.
      v_fiscweek = i_fiscwk.
  ENDP.

* Update the DTP with prior fiscal week.
  DATA: v_fascid TYPE string,
    v_fiscal_week TYPE string.
    v_fascid = i_fascid.
    v_fiscal_week = v_fiscweek.
    CALL FUNCTION 'Y_CHANGE_DTP_FILTERS'
      EXPORTING
        p_dtp = i_dtp
        p_fla0b1 = i_fascid
        p_fiscal = i_fiscwk
        p_optimal = 'EQ'.
```
End result:

On the 1st day of every week DTP loads a snapshot of the previous calendar week data via a scheduled process chain load. Using the custom ABAP program approach right DTP execute variant, we need an ABAP program variant to update DTP fiscal week filter with previous fiscal week value.
Related Content

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.