Step by Step Guide to Creating a Process Type to Close an Open Request in a Cube in BI 7.0

Applies to:

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

Summary

You want to create a Process Type to close an open Request in a Cube. At present, there is no standard Process Type available in SAP Netweaver 2004s that offers this functionality.

Author: Nageswara Reddy M

Company: Mahindra Satyam Computer Services Ltd.

Created on: 03 August 2009

Author Bio

Nageswara Reddy is currently working with Mahindra Satyam Computer Services Ltd. He has an overall experience of 6 years in SAP- ABAP. He has been working in SAP-BI 7.0 for 2 years now.

Table of Contents

Introduction

What exactly does the term "Close open request in Infocube" mean?

Closing the request means making the request available for reporting.

This is done in the case of planning cubes, also called Transactional cubes / Realtime cubes.

When a request is being loaded into a transactional cube, the request is flagged as ‘open’ (Yellow QM status) until it receives 50,000 records (from user planning). Once a request crosses 50,000 records, it is considered Closed, the QM status is turned to green and the data in it becomes available for reporting. If this data needs to be reported on before 50,000 records are added, the request needs to be manually closed (make it green) so that the records are available for reporting.

Generally to close the requests in a cube, the program RSAPO_CLOSE_TRANS_REQUEST_ALL3 can be used. When this program is executed, it closes all open requests in all the transactional cubes in the system. But the user may want to close the request his relevant cube alone while leaving the open requests in other cubes untouched. So the user has to create a program using the code in this program or by making use of the Function Module RSAPO_CLOSE_TRANS_REQUEST. This program and a variant saved with the infocube details can be used in the ABAP process type.

The other option to close open requests is by using the Standard Process type 'Switch Realtime InfoCube to Load Mode'. This will close the request and at the same time switch the Realtime cube to Load mode. So if some other user is trying to write some data to the same cube simultaneously, it will throw an error.

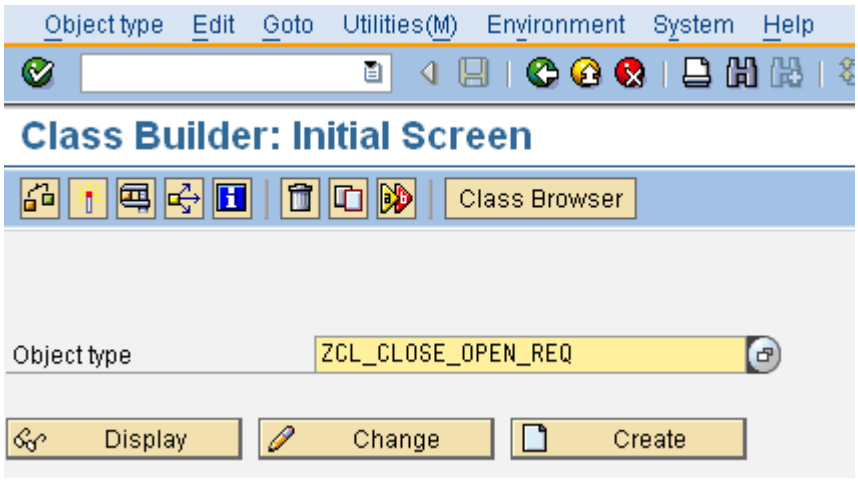
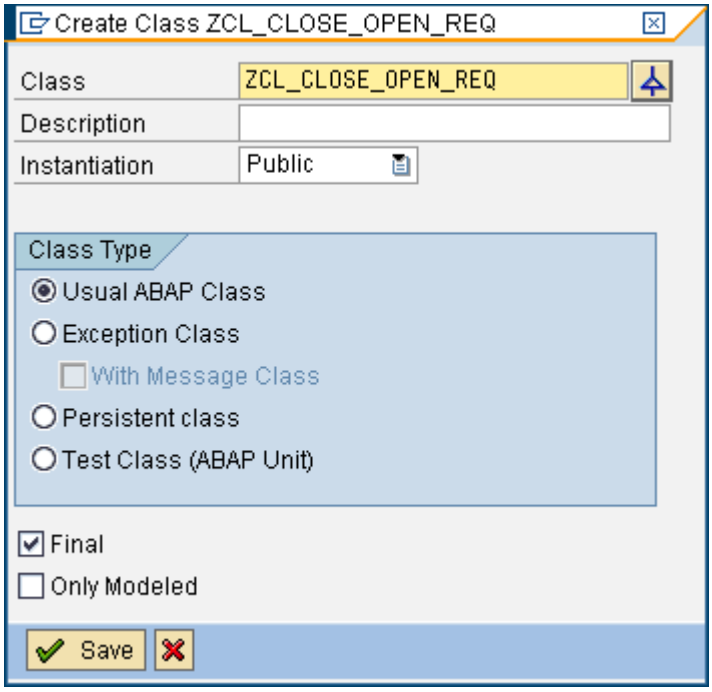
The following criteria need to be considered while creating a Process type.

1. That is user friendly compared to the ABAP process type.
2. That closes the open request without switching the cube to Load mode.

1. Step by Step Solution

2.1 Implementation of ABAP OO class for process type

The coding for the implemented methods is provided in the Appendix.

Enter the Class builder via SE24. Specify the name of the class and press the 'Create' button.	
Choose the Class Type as Usual ABAP Class.	

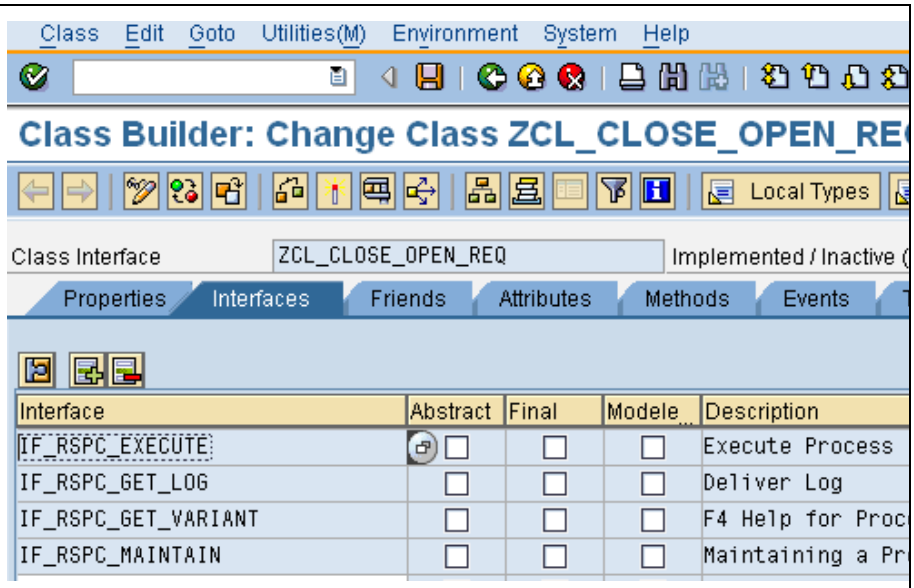
Switch to the 'Interfaces' tab and specify the following interfaces:

- IF_RSPC_EXECUTE
Mandatory interface, which is called on the execution of the process type

- IF_RSPC_GET_LOG
Returns the log for a variant and an instance to the process chain maintenance

- IF_RSPC_GET_VARIANT
Optional interface for the selection of the variants

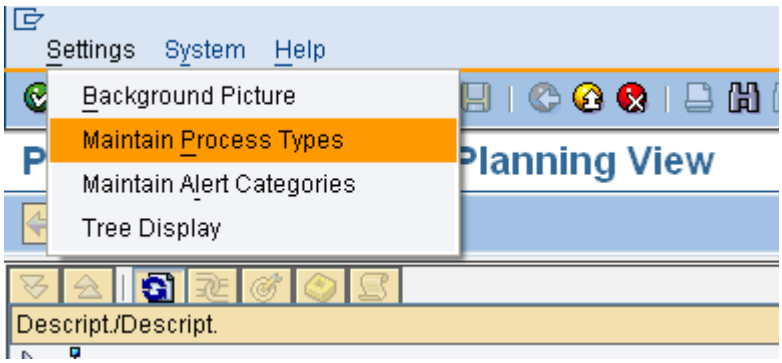
- IF_RSPC_MAINTAIN
Optional interface for the maintenance of the variants (in our case call of the Process Chain maintenance)



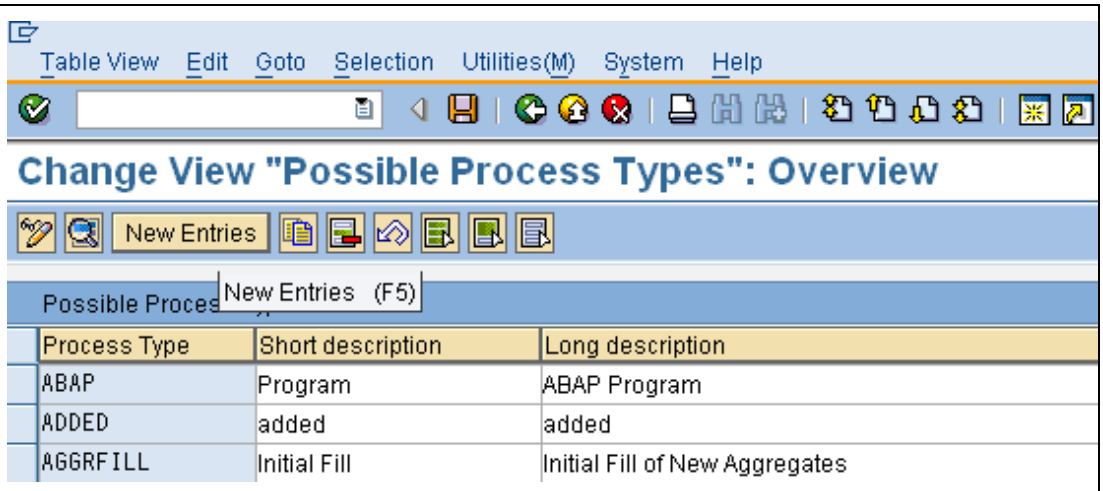
2.2 Create & specify setting for new Process Type

After you have implemented the class that performs the processing and maintenance of your custom- defined process type, you have to make the new process type known in the process chain maintenance.

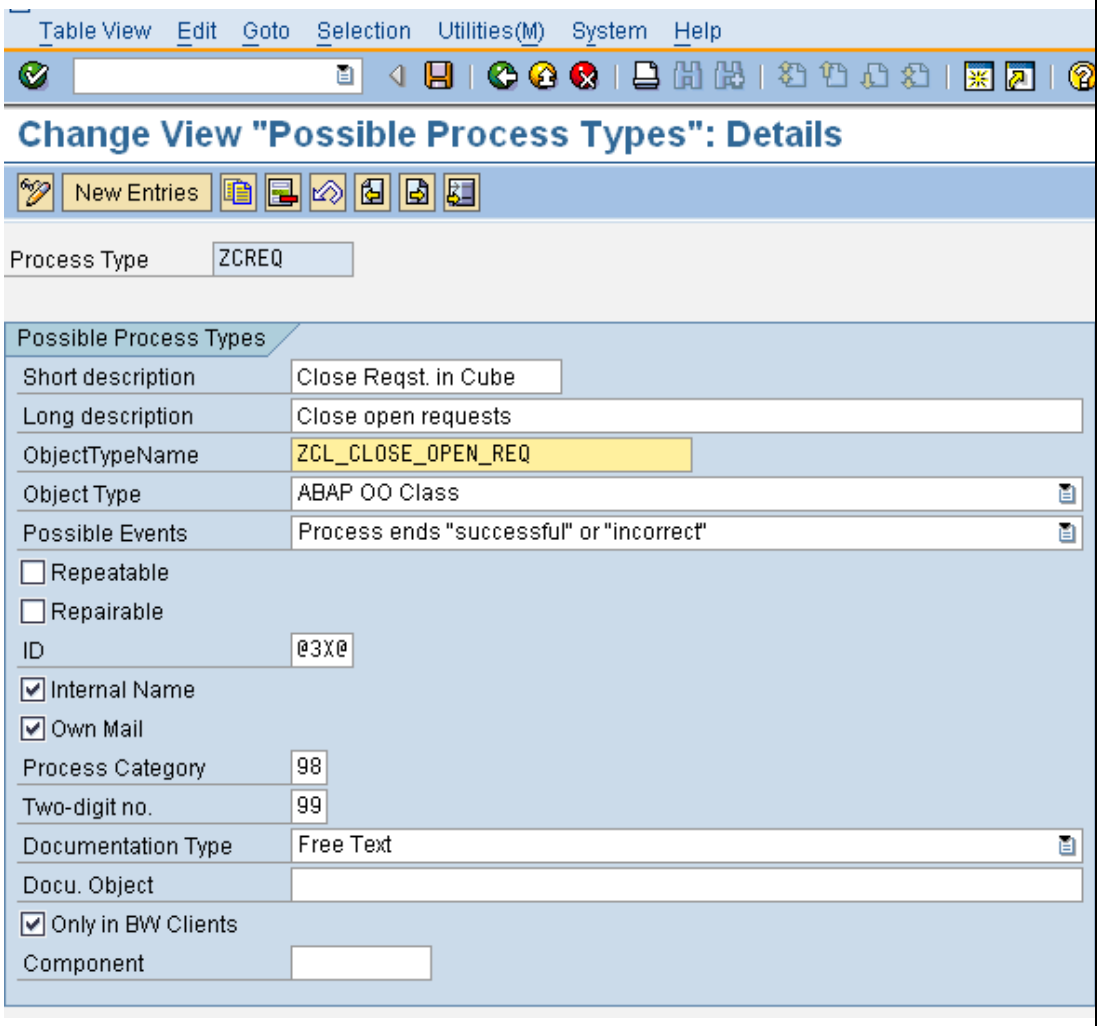
In Transaction RSPC, go to 'Maintain Process Types' via the Menu bar.

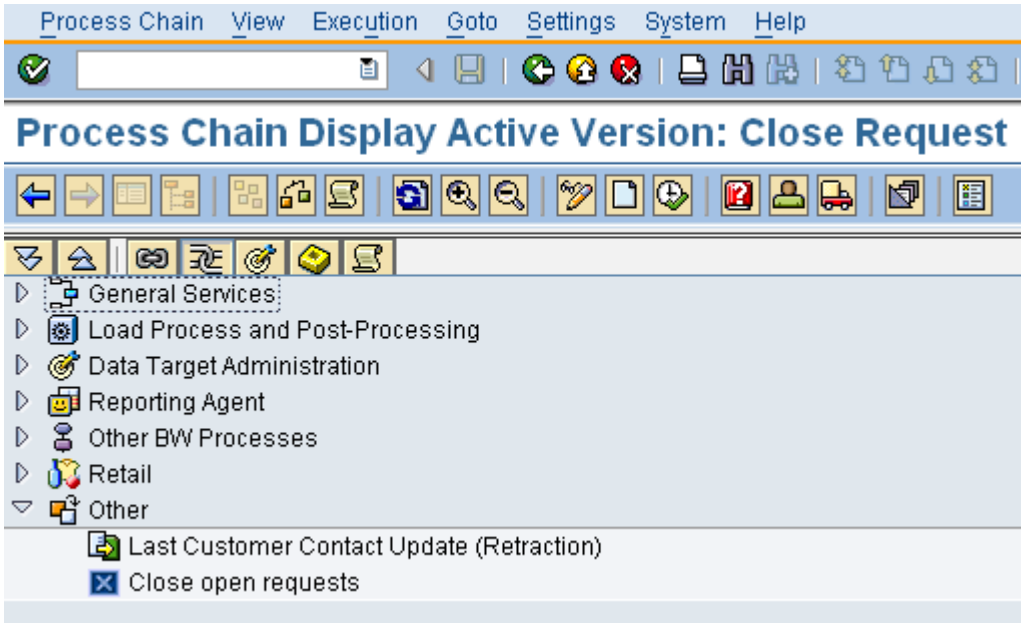


In the following screen, click on the New Entries Button.



Here give a name for the New Process type (ZCREQ for example). Give the name of the class in ObjectTypeName .



Now you can find the new process type underneath the category specified. In this example it is 'Other'.	

3. Appendix

The appendix provides the coding for the implemented methods. You can make any further changes as per your requirements.

```
IF_RSPC_MAINTAIN~MAINTAIN
METHOD if_rspc_maintain~maintain.

  MESSAGE s002(sy) WITH 'Change/Create Options Disabled'.

ENDMETHOD.

IF_RSPC_GET_VARIANT~GET_VARIANT
METHOD if_rspc_get_variant~get_variant.

  TYPES: BEGIN OF ty_infocube.
  TYPES: infocube TYPE rsinfocube,
         text     TYPE rstxtlg,
         END OF ty_infocube.

  DATA: lt_infocube TYPE TABLE OF ty_infocube,
        la_infocube type ty_infocube,
        lt_rsdcubet TYPE TABLE OF rsdcubet,
        la_rsdcubet type rsdcubet,
        lt_rsdcube  TYPE TABLE OF rsinfocube,
        la_rsdcube  TYPE rsinfocube,
        lt_retval   TYPE TABLE OF ddshretval,
        la_retval   TYPE ddshretval.

  SELECT *
    FROM rsdcubet
    INTO TABLE lt_rsdcubet
   WHERE objvers = 'A'.

  SELECT infocube
    INTO TABLE lt_rsdcube
    FROM rsdcube
   WHERE transact = 'X'
     AND objvers = 'A'.

  LOOP AT lt_rsdcube INTO la_rsdcube.
    la_infocube-infocube = la_rsdcube.
    READ TABLE lt_rsdcubet INTO la_rsdcubet WITH KEY infocube = la_rsdcube.
    IF sy-subrc = 0.
      la_infocube-text = la_rsdcubet-txtlg.
    ENDIF.
    APPEND la_infocube TO lt_infocube.
  ENDLOOP.

  CALL FUNCTION 'F4IF_INT_TABLE_VALUE_REQUEST'
    EXPORTING
      retfield        = 'INFOCUBE'
      window_title    = 'Select the Cube'
      value_org       = 'S'
      display         = rs_c_false
    TABLES
```

```
      value_tab       = lt_infocube
      return_tab      = lt_retval
    EXCEPTIONS
      parameter_error = 1
      no_values_found = 2.

  READ TABLE lt_retval INTO la_retval INDEX 1.
  IF sy-subrc EQ 0.
    e_variant = la_retval-fieldval.
    READ TABLE lt_infocube INTO la_infocube WITH KEY infocube = la_retval-fieldval.
    IF sy-subrc = 0.
      e_variant_text = la_infocube-text.
    ENDIF.
  ENDIF.

ENDMETHOD.

IF_RSPC_GET_VARIANT~WILDCARD_ENABLED
METHOD if_rspc_get_variant~wildcard_enabled.

  result = rs_c_false.

ENDMETHOD.

IF_RSPC_GET_VARIANT~EXISTS
METHOD if_rspc_get_variant~exists.

  r_exists = rs_c_true.

ENDMETHOD.

IF_RSPC_GET_LOG~GET_LOG
METHOD if_rspc_get_log~get_log.

  IF NOT i_instance IS INITIAL.
    CALL METHOD cl_rspc_appl_log=>display
      EXPORTING
        i_type           = rspc1_c_type-custexit
        i_variant        = i_variant
        i_instance       = i_instance
        i_no_display     = 'X'
      IMPORTING
        e_t_msg          = e_t_messages
      EXCEPTIONS
        internal_failure = 1
        OTHERS           = 2.
  ENDIF.

ENDMETHOD.


IF_RSPC_EXECUTE~EXECUTE
METHOD if_rspc_execute~execute.

  FIELD-SYMBOLS: <f_rsapoadm>   TYPE rsapoadm.
```

```
  DATA: lt_rsapoadm  TYPE TABLE OF rsapoadm,
        l_subrc      TYPE i,
        l_r_infocube TYPE REF TO cl_rsdri_infocube_realtime,
        l_uid        TYPE sysuuid_25,
        l_timestamp  TYPE rstimestmp.

  DATA: l_r_appl_log  TYPE REF TO cl_rspc_appl_log,
        ls_msg       TYPE rs_s_msg.

* Get timestamp
  CALL FUNCTION 'RSSM_GET_TIME'
    IMPORTING
      e_timestamps = l_timestamp.

* Get Instance
  CALL FUNCTION 'RSSM_UNIQUE_ID'
    IMPORTING
      e_uni_idc25 = l_uid.
  MOVE l_uid TO e_instance.

* get list of all relevant infocubes
  SELECT *
    INTO TABLE lt_rsapoadm
    FROM rsapoadm
   WHERE infocube  EQ i_variant
     AND wr_reqsid <> 0.

  IF lt_rsapoadm[] IS NOT INITIAL.

* close open requests
    LOOP AT lt_rsapoadm ASSIGNING <f_rsapoadm>.

      CREATE OBJECT l_r_infocube
        EXPORTING
          i_infocube = <f_rsapoadm>-infocube.

      CALL METHOD l_r_infocube->current_request_close
        EXCEPTIONS
          illegal_input      = 1
          request_not_closed = 2
          OTHERS             = 4.

      l_subrc  = sy-subrc.

      CLEAR   ls_msg-msgid.
      ls_msg-msgid = 'ZWDW0001_MESSAGE'.
      CASE l_subrc.
*   success
        WHEN 0.
          ls_msg-msgty = 'S'.
          ls_msg-msgno = '000'.
          ls_msg-msgv1 = <f_rsapoadm>-wr_reqsid.
          ls_msg-msgv2 = <f_rsapoadm>-infocube.
          e_state = 'G'.
*   not closed
        WHEN 2.
```

```
          ls_msg-msgty = 'W'.
          ls_msg-msgno = '001'.
          ls_msg-msgv1 = <f_rsapoadm>-wr_reqsid.
          ls_msg-msgv2 = <f_rsapoadm>-infocube.
          e_state = 'R'.
*   other error
        WHEN OTHERS.
          ls_msg-msgty = 'E'.
          ls_msg-msgno = '002'.
          ls_msg-msgv1 = <f_rsapoadm>-wr_reqsid.
          ls_msg-msgv2 = l_subrc.
          ls_msg-msgv3 = <f_rsapoadm>-infocube.
          e_state = 'R'.
      ENDCASE.
    ENDLOOP.
  ELSE.
    ls_msg-msgty = 'I'.
    ls_msg-msgno = '003'.
    ls_msg-msgid = 'ZWDW0001_MESSAGE'.
    ls_msg-msgv1 = i_variant.
    e_state = 'G'.
  ENDIF.

  CALL METHOD cl_rspc_appl_log=>create
    EXPORTING
      i_type       = rspc1_c_type-custexit
      i_variant    = i_variant
      i_instance   = e_instance
    RECEIVING
      r_r_appl_log = l_r_appl_log.

  CONCATENATE '' i_variant INTO ls_msg-msgv2.
  CALL METHOD l_r_appl_log->add_message
    EXPORTING
      i_msgty = ls_msg-msgty
      i_msgid = ls_msg-msgid
      i_msgno = ls_msg-msgno
      i_msgv1 = ls_msg-msgv1
      i_msgv2 = ls_msg-msgv2
      i_msgv3 = ls_msg-msgv3.

  CALL METHOD l_r_appl_log->save
    EXCEPTIONS
      internal_failure = 1
      OTHERS           = 2.

  FREE l_r_appl_log.

  IF e_state IS INITIAL.
    e_state = 'G'.
  ENDIF.

ENDMETHOD.
```

Related Content

https://forums.sdn.sap.com/thread.jspa?messageID=4065094#4065094

https://forums.sdn.sap.com/thread.jspa?messageID=4411885#4411885

https://forums.sdn.sap.com/thread.jspa?messageID=4936550#4936550

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.