FI - Auto Email Notification for Closing Cockpit

Applies to:

SAP ECC 6.0

Summary

The closing process is one of the most important business processes in financial accounting. This article illustrates the setting of auto email notification using SAP enhancement framework in this process. It describes how SAP HTML events can be handled to send auto notifications to the business users about the status of the tasks.

Author: Abhinav Sharma

Company: HCL Technologies Ltd

Created on: 27 June 2011

Author Bio

Abhinav Sharma is SAP certified NetWeaver Consultant. His area of expertise includes ABAP, Webdynpro Java/ABAP and SAP EP. .

Table of Contents

Process Overview

One of the most complex processes in *Financial Accounting* is the *Closing Process*. The closing process is the recurring process that can occur daily, monthly and yearly. The Closing Cockpit (Transaction Code – CLOCO) can be use to simplify the complex processes of financial closing by providing transactions and programs.

It is the set of event-driven activities that optimizes the business processes. It provides monitoring and analysis tools that covers entire closing process.

Following structural objects are required for Closing Cockpit.

1. ***Hierarchies*** to display the organizational objects involved in the closing process
2. A ***task list template*** based on the organizational structure
3. A detail view of the ***characteristic values*** of the individual hierarchy levels used in the task list template.
4. ***Task lists*** that are derived from the task list template
5. A ***list display*** in which all tasks to be managed or executed from the respective task list are made available for processing or for monitoring task progress
6. A ***monitor*** that shows a graphical representation of the critical paths as well as the processing periods and processing sequence with their respective dependencies
7. ***Detailed information*** about the technical settings of tasks as well as for analyzing background programs (spool, job log information)
8. ***Dependencies*** for displaying the conditions representing a prerequisite for processing the individual tasks

To know more about these structured objects, refer Configuring the Closing Cockpit.

Business Requirement

The closing process is one of the most important processes in financial accounting. CLOCO transaction is used to execute closing cockpit programs. The user wants to receive the email notification of the status of the task performed. The notification by email should be reached to the user if the task is ended with warnings or with or without errors. It is required to find out a way to send these email notifications whenever there is any change in the status of the task executed. This task can be executed manually or can be schedule in a background job.

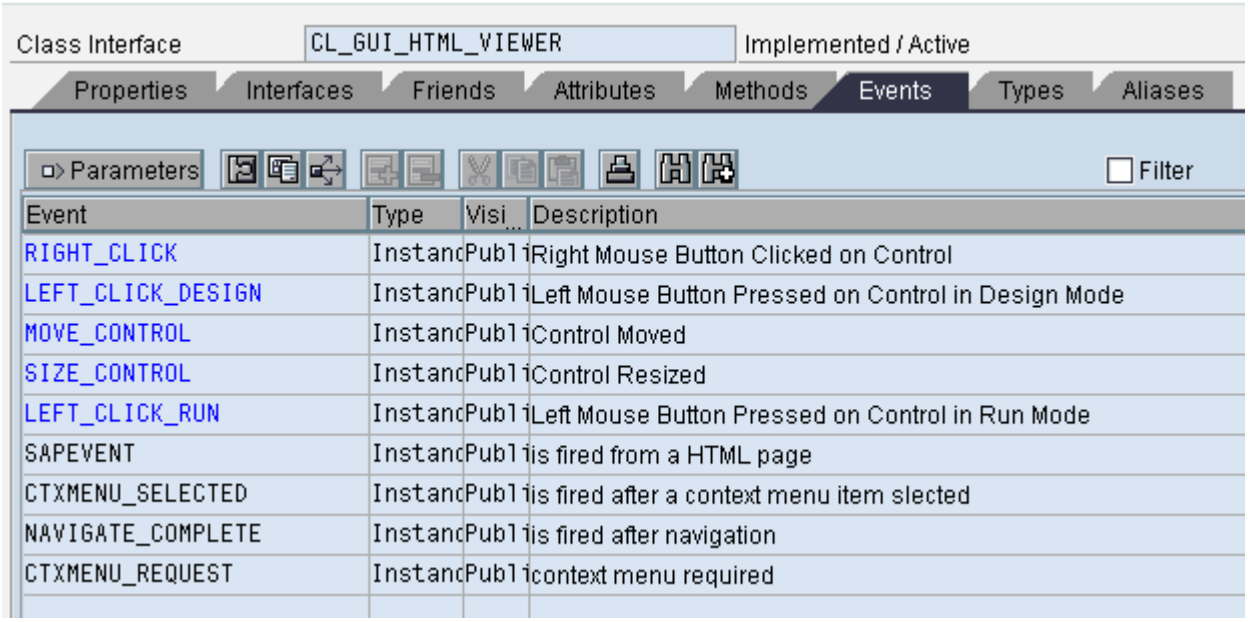
Technical Solution

When user executes the CLOCO or CLOCOC certain events are raised. These events can be put into use to send the email notification to the user. Closing Cockpit uses SAP HTML Viewer for displaying task list and its corresponding details. SAP HTML Viewer is a control developed by SAP and it can be used with other controls like buttons or list boxes on a screen in an R/3 transaction. CLOCO/CLOCOS transactions use these controls.

One of the classes that are being used in CLOCO/CLOCOS transactions is **`CL_GUI_HTML_VIEWER`**. This class has an event by name **`SAPEVENT`**. This event is triggered whenever an event is defined on the HTML page with type **`SAPEVENT`** is triggered by the user. Use of these events on the HTML page is to control the objects. The event that gets triggered under **`SAPEVENT`** is of the following format

SAPEVENT:<action>?data.

Where **`action`** is the name of the action performed on SAP HTML page and **`data`** is the data passed along with the **`SAPEVENT`**. **`SAPEVENT`** is defined in class **`CL_GUI_HTML_VIEWER`** as described below.



This event returns the following parameters:

Parameters	Descriptions
action	Action parameter of the SAPEVENT
frame	Name of the HTML frame in which event is triggered
getdata	Data in the event string
postdata	Data table for post data
query_table	Event table in the form

Using this event we can get the corresponding action and then can build logic of automating the email notifications by enhancing the respective method.

The challenge is to identify the method which can be enhanced and can provide the data corresponding to the HTML frame.

The **`DISPATCH`** method of **`CL_GUI_HTML_VIEWER`** is used to dispatch the application events to the registered event handlers of the event. If not called explicitly, it will be called by the system after the PAI is processed. Important point to remember is that an event can be dispatched only once. Any attempt to dispatch the events second time does not trigger the handler events again.

On **`Dispatch`** method of **`CL_GUI_HTML_VIEWER`**, following even is raised

```
RAISE EVENT sapevent
  EXPORTING
    action      = m_action
    frame       = m_frame
    getdata     = m_getdata
    postdata    = m_postdata
    query_table = m_query_table.
```

The event raised will pass the action name, frame, getdata, postdata and a query_table to the **`SAPEVENT`**. For typical scenario that results in warning, the value of m_postdata can be as follow

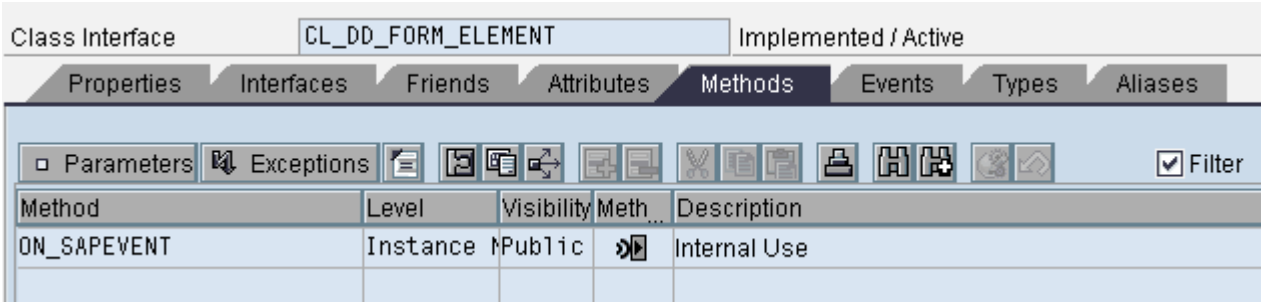
Table	M_POSTDATA
Table Type	Standard Table[1x1(512)]

Line	TABLE_LINE[C(256)]
1	sapbu_cl= &sapse_cl=A1T2C2F1S1&sapin_cl= &A1T2C2F1S1=DD_SEL_ERROR

The query table will contain the data as shown below if the status results in **ERROR**

Table	M_QUERY_TABLE	
Table Type	Standard Table[4x2(560)]	
Line	NAME[C(30)]	VALUE[C(250)]
1	sapbu_cl	
2	sapse_cl	A1T2C2F1S1
3	sapin_cl	
4	A1T2C2F1S1	DD_SEL_ERROR

Once the event is raised, the control goes to the registered event handler. For **SAPEVENT**, there is an event handler name **ON_SAPEVENT** in class **CL_DD_FORM_ELEMENT** is available.



This handler identifies the current object, current element and query. It gets the ID of the clicked button/select/input element and processes query. This handler raises another event on item selected by exporting the ID of the element and that is handled by **ON_SELECTED** handler of class **CL_DD_SELECT_ELEMENT**. **ON_SELECTED** event handler raises a **select** event which is handled by **HANDLE_DD_SELECT_CLICKED** event handler of the class **CL_DD_DOCUMENT_FC**. This method sets the *p_status* value based on following logic.

```
CASE sender->value.
  WHEN 'DD_SEL_FINISHED'.
    p_status = '0'.
  WHEN 'DD_SEL_WARNING'.
    p_status = '2'.
  WHEN 'DD_SEL_ERROR'.
    p_status = '4'.
  WHEN OTHERS.
ENDCASE.
```

The value of *p_status* tells about the status of the task selected and this value along with the other values in the HTML frame is passed to **ON_ACTION** method of class **CL_ACTION_MANAGER** (see below code snippet).

```
CALL METHOD cl_action_manager=>on_action
  EXPORTING
    id_do_raise_event = 'X'
    ir_item           = lr_item
    id_action         = cl_action_manager=>cmd_act_status_change
```

```
id_do_lock        = 'X'
id_value          = p_status
ir_cfc            = m_cfc.
```

CL_ACTION_MANAGER class is responsible for data manipulation and eventing of HTML frame.

Enhancement Code

To automate the email notification, create an enhancement implementation in method ON_ACTION. Using following code, the notifications can be sent to the set of recipients.

```
//This is a code sample block
*** Enhancement to send email notification

IF sy-tcode = 'CLOCO' OR sy-tcode = 'CLOCOC' AND lr_item IS NOT INITIAL.
* Data for send function
  DATA: DOC_DATA  type SODOCCHGI1,
        OBJECT_ID type SOODK,
        wa_OBJCONT   type SOLI,
        OBJCONT   type STANDARD TABLE OF SOLI,
        wa_RECEIVER  type SOMLRECI1,
        RECEIVER  type STANDARD TABLE OF SOMLRECI1.

** Recipient Determination
  DATA: lt_email type STANDARD TABLE OF zemail,
        wa_email type zas_email.

   SELECT firstname lastname emailid FROM zemail
          INTO CORRESPONDING FIELDS OF TABLE lt_email WHERE isactive = 'X'.

   LOOP AT lt_email into wa_email.

          wa_RECEIVER-receiver = wa_email-emailid.
          TRANSLATE wa_RECEIVER-receiver TO LOWER CASE.
          wa_RECEIVER-rec_type = 'U'.
          APPEND wa_RECEIVER to RECEIVER.

   ENDLOOP.

** Mail Subject
   DOC_DATA-OBJ_DESCR = LR_ITEM->M_TEXT.


** Mail Content Body
   DATA:  msg type string,
          userstatus type SCHEDMAN_JOB_STATI,
          lt_smoutput type  SMOUTPUT.

   APPEND space to objcont.

   CONCATENATE 'TASK : ' LR_ITEM->M_TEXT INTO msg SEPARATED BY SPACE.
   wa_OBJCONT-LINE = msg.
   APPEND wa_OBJCONT to objcont.
   clear msg.

   APPEND space to objcont.
```

```
   CALL METHOD LR_ITEM->GET_SMOUTPUT
   IMPORTING
               ES_OUTPUT = lt_smoutput.

   userstatus = lt_smoutput-USERSTAT.

   CONCATENATE 'STATUS OF THE TASK : ' lt_smoutput-USERSTAT_TEXT INTO msg SEPARATED
BY SPACE.
          wa_OBJCONT-LINE = msg.
          APPEND wa_OBJCONT to objcont.
          clear msg.

** Send Email notification to recipients
   CALL FUNCTION 'SO_NEW_DOCUMENT_SEND_API1'
          EXPORTING
           DOCUMENT_DATA              = DOC_DATA
           DOCUMENT_TYPE = 'RAW'
           PUT_IN_OUTBOX = 'X'
           COMMIT_WORK   = 'X'
          IMPORTING
           NEW_OBJECT_ID              = OBJECT_ID
          TABLES
           OBJECT_CONTENT             = OBJCONT
           RECEIVERS                  = RECEIVER
          EXCEPTIONS
           TOO_MANY_RECEIVERS         = 1
           DOCUMENT_NOT_SENT          = 2
           DOCUMENT_TYPE_NOT_EXIST    = 3
           OPERATION_NO_AUTHORIZATION = 4
           PARAMETER_ERROR            = 5
           X_ERROR                    = 6
           ENQUEUE_ERROR              = 7
           OTHERS                     = 8.

   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.
```

Once the user executes the transaction CLOCO/CLOCOC and do the change, email notification will be sent to the user. This can be checked using transaction SOST

References

To explore more about Closing Cockpit and SAP HTML Viewer, kindly refer following topics

Configuring the Closing Cockpit

Closing and Reporting

SAP HTML Viewer

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.