SAP Records Management

Starting a Workflow in a Record by Entering Import Parameters

May 4, 2004

Contents

1 Introduction

The following tutorial is aimed at consultants who are implementing Records Management. Prerequisites are a working knowledge of Records Management, SAP Business Workflow, and ABAP objects.

Customers often want user workflows to be triggered directly from a record. This tutorial shows you a user-friendly way of meeting this requirement.

Take the following scenario: you want a record to start a workflow in which the record itself is sent to another employee. The user who starts the workflow needs to see a dialog box in which he or she can enter the employee who receives the record. The user also needs to enter a deadline by which the recipient of the record must process it.

This can be realized as follows: the service provider for workflows provides the connection parameter FUNCTION_START_WORKFLOW. You can set this parameter to the name of a function module that starts the workflow. We want to use this connection parameter and implement a function module that calls a dialog box for entering the import parameters and starts the workflow.

2 Prerequisites

You have created a workflow template that enables the record to be forwarded to another user while specifying a deadline. The workflow template has the following input parameters:

- User: This parameter must be typed as a DDIC reference, such as T77UA-UNAME.
- Deadline: This parameter must be typed as a DDIC reference, such as SWWWIDEADL-WI_LED.
- Record: The technical name of the parameter must be RECORD, and the parameter must have the object type RECORD.

You integrate a standard task into the workflow template. This task calls the RECORD.DISPLAY method.

3 Setting Up an Element Type in the Service Provider for Workflows

Create a new element type in the service provider for workflows. Make the following entries on the *Connection Parameter* tab page:

- Connection parameter FUNCTION_START_WORKFLOW: name of a new function module for starting the workflow, to be implemented by you (see below)
- Connection parameter WF_DEFINITION: ID of the workflow template that you want to execute (see above)
- Connection parameters RFC_DESTINATION and RFC_DESTINATION_CALLBACK: ‘NONE’

4 Implementing a Function Module for the Connection Parameter FUNCTION_START_WORKFLOW

The standard SAP system already includes the function module SRM_SP_WFL_START_WORKFLOW, which you can enter for the connection parameter FUNCTION_START_WORKFLOW. The interface of your own function module must match the interface of the standard function module. To guarantee this, copy the standard function module SRM_SP_WFL_START_WORKFLOW to your own function module.

4.1 Differences from the Standard Function Module

We do not use the standard function module since it triggers a dialog box for entering import parameters that is not very user-friendly.

You use the standard function module on ly if the input parameters set by the user have complex data types (for example, business object types, structures, or tables) and the workflow is started typically by an administrator. If the input parameters of the workflow are scalar parameters that are typed as ABAP Dictionary references, then we recommend that you implement a separate function module. You can adopt some parts of the standard function module when you do this.

For more information about the standard function module, see the *Records Manag ement* documentation under *Customizing → Service Provider for SAP Business Workflows*.

4.2 Calling a Dialog for Entering Import Parameters

To call the dialog box for entering the import parameters, use the POPUP_GET_VALUES function module. When you specify DDIC references, this function module provides you with a screen that includes F1 help and F4 help.

In the FIELDS parameter of the POPUP_GET_VALUES function module, you have to specify the DDIC references of the fields you want to display. Before you do this, use the definition of the workflow container to see which import parameters have a DDIC reference. Extract the DDIC information (`tabname`, `fieldname`) and send it to the POPUP_GET_VALUES function module. The function module now generates the input dialog box with the appropriate fields, field labels, F1 help, and F4 help (if they exist). However, the input is not tested.

In our case, *User Name* (for work item recipients) and *Latest End* are displayed as input fields.

4.3 Example Code

The following is an example of the code for a function module that starts the workflow.

Notes:

- Most parts of this example are taken from the standard function module SRM_SP_WFL_START_WORKFLOW, but we have given them new comments to make them easier to understand. The new parts have bee n given the comments *Enhancement BEGIN* and *Enhancement END*.
- This example has been implemented in WebAS 6.20, and is only guaranteed to be valid for this release.

```
FUNCTION Z_SRM_START_WORKFLOW_DEMO.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     REFERENCE(TASK) TYPE  SWW_TASK
*"     REFERENCE(RFC_DESTINATION) TYPE  RFCDEST
*"     REFERENCE(SAP_RELEASE) TYPE  RFCSI-RFCSAPRL
*"     REFERENCE(INPUT_CONTAINER) TYPE  SWFNAMVTAB
*"  EXPORTING
*"     REFERENCE(RETURN_CODE) TYPE  SYSUBRC
*"     REFERENCE(WORKITEM_ID) TYPE  SWW_WIID
*"     REFERENCE(CANCELLED) TYPE  XFLAG
*"  TABLES
*"      RETURN STRUCTURE  BAPIRET2 OPTIONAL
*"  EXCEPTIONS
*"      INTERNAL_ERROR
*"      REMOTE_NOT_SUPPORTED
*"----------------------------------------------------------------------

  TYPE-POOLS: swfcn, srmw2, swfex.

  DATA:
```

```
  container_ref                 TYPE REF TO if_swf_cnt_container,
  my_iterator                   TYPE REF TO if_swf_cnt_iterator,
  my_element                    TYPE REF TO if_swf_cnt_element,
  prop_import                   TYPE xstring,
  prop_system                   TYPE xstring,
  prop_optional                 TYPE xstring,
  prop_null                     TYPE xstring,              "#EC NEEDED
  prop_initial                  TYPE xstring,              "#EC NEEDED
  wa_return                     LIKE LINE OF return,
  errors                        TYPE swft100tab,           "#EC NEEDED
  warnings                      TYPE swft100tab,           "#EC NEEDED
  num_elements_imported         TYPE int4.                 "#EC NEEDED

********* Set properties *************
  prop_import   = swfcn_p_param_import.
  prop_system   = swfcn_p_system.
  prop_optional = swfcn_p_param_optional.
  prop_null     = swfcn_p_is_null.
  prop_initial  = swfcn_p_is_initial.

********* Read workflow container ***********
  CLEAR: return_code, return, container_ref.
  CALL FUNCTION 'SRM_WAPI_READ_CONTAINER'
    EXPORTING
      rfc_destination        = rfc_destination
      sap_release            = sap_release
      task                   = task
    IMPORTING
      return_code            = return_code
      container_ref          = container_ref
    TABLES
      return                 = return.

  IF NOT return_code IS INITIAL.
    RAISE internal_error.
  ENDIF.

  IF NOT container_ref IS BOUND.
    return_code = 999.
    CLEAR: return, wa_return.
    wa_return-message = text-001.
    wa_return-system  = rfc_destination.
    APPEND wa_return TO return.
    RAISE internal_error.
  ENDIF.

********* Fill key of BO RECORD ***********
  DATA: wa_input_container    TYPE swaconextv,
        my_por                TYPE swotobjid, "persistent object reference
        my_input_container    TYPE swfnamvtab,
        lrh_element           TYPE REF TO if_swf_cnt_element,
        test_objtype          TYPE string,
        test_objkind          TYPE swfobjkind.

  DATA: BEGIN OF bo_record_key,
            docclass   TYPE bapisrmrec-docclass,
            objectid   TYPE bapisrmrec-objectid,
        END OF bo_record_key.

  CLEAR my_input_container.
  my_input_container = input_container.                    "1:1 Copie

  TRY.

** Test whether container element RECORD is business object RECORD
      CLEAR: lrh_element, test_objtype, test_objkind.
      lrh_element = container_ref->get_element_def( name = 'RECORD' ).
      IF NOT lrh_element IS INITIAL.
        CALL METHOD lrh_element->get_type
          IMPORTING
            objtype = test_objtype
            objkind = test_objkind.
      ENDIF.

      IF test_objtype = 'RECORD' AND test_objkind = 'BO'.

** Fill key of BO RECORD and append to my_input_container
        CLEAR: wa_input_container, bo_record_key.
        LOOP AT input_container INTO wa_input_container.
          CASE wa_input_container-element.
            WHEN srmw2_con_aktcl.
              bo_record_key-docclass = wa_input_container-value.
            WHEN srmw2_con_aktid.
              bo_record_key-objectid = wa_input_container-value.
          ENDCASE.
        ENDLOOP.

        CLEAR my_por.

        my_por-objtype    = 'RECORD'.
        my_por-objkey    = bo_record_key.
        my_por-describe  = swfex_bor_por_describe.

        wa_input_container-element = 'RECORD'.
        wa_input_container-value   = my_por.
        APPEND wa_input_container TO my_input_container.

      ENDIF.

    CATCH cx_swf_cnt_container. "do nothing (no append to my_input_container)
  ENDTRY.


*********  Fill container with input container ("merge") *********
  CLEAR: errors, warnings, num_elements_imported.
  CALL METHOD container_ref->import_from_simple_container
    EXPORTING
      values                = my_input_container
      import_param          = 'X'
      export_param          = space
      changing_param        = space
      returning_param       = space
      suppress_others       = 'X'
      existence_check       = space
      no_system_elements    = space
      undefined_handling    = space
    IMPORTING
      errors                = errors
      warnings              = warnings
      num_elements_imported = num_elements_imported.

  IF NOT container_ref IS BOUND.
    return_code = 999.
    CLEAR: return, wa_return.
    wa_return-message = text-006.
    wa_return-system  = rfc_destination.
    APPEND wa_return TO return.
    RAISE internal_error.
  ENDIF.

********* Enhancement BEGIN ***********
  DATA: get_value_wa TYPE sval,
        get_value_tab TYPE TABLE OF sval,
        el_ddic TYPE dfies,
        imp_elements TYPE TABLE OF REF TO if_swf_cnt_element.

  CLEAR: my_iterator, my_element.

** Read DDIC information of import container elements
  CALL METHOD container_ref->get_iterator
    EXPORTING
      im_properties_x         = prop_import
      im_exclude_properties_x = prop_system
    IMPORTING
      ex_iterator             = my_iterator
      ex_element              = my_element.

  WHILE NOT my_element IS INITIAL.
    CLEAR el_ddic.
    CALL METHOD my_element->get_repository_typeinfo
      IMPORTING
        ddic_info = el_ddic.

    IF NOT el_ddic IS INITIAL.
      get_value_wa-tabname   = el_ddic-tabname.
      get_value_wa-fieldname = el_ddic-fieldname.
      APPEND get_value_wa TO get_value_tab.
      APPEND my_element TO imp_elements.
    ENDIF.
    my_element = my_iterator->get_next( ).

  ENDWHILE.

** Call Popup
  CALL FUNCTION 'POPUP_GET_VALUES'
    EXPORTING
*     NO_VALUE_CHECK        = ' '
      popup_title           = text-val
*     START_COLUMN          = '5'
*     START_ROW             = '5'
   IMPORTING
     returncode            = cancelled
    TABLES
      fields                = get_value_tab
   EXCEPTIONS
     ERROR_IN_FIELDS       = 1
     OTHERS                = 2.

  IF cancelled IS INITIAL.
    LOOP AT imp_elements INTO my_element.
      READ TABLE get_value_tab INTO get_value_wa INDEX sy-tabix.
      TRY.
          CALL METHOD my_element->set_value
            EXPORTING
              value = get_value_wa-value.
        CATCH cx_swf_cnt_elem_not_found .
        CATCH cx_swf_cnt_elem_access_denied .
        CATCH cx_swf_cnt_element .
        CATCH cx_swf_cnt_container .
      ENDTRY.
    ENDLOOP.
  ELSE.
    return_code = 999.
    CLEAR: return, wa_return.
    wa_return-message = text-007.
    wa_return-system  = rfc_destination.
    APPEND wa_return TO return.
    RAISE internal_error.
  ENDIF.
********* Enhancement END ****************
```

```
***** Don't start workflow remotely when its container still has empty obligatory
** elements
  IF NOT rfc_destination IS INITIAL AND rfc_destination NE 'NONE'.
** ==> workflow runs in a remote system

** Does container still have empty obligatory elements?
    CLEAR: my_iterator, my_element.
    CALL METHOD container_ref->get_iterator
      EXPORTING
        im_exclude_properties_x = prop_optional
      IMPORTING
        ex_iterator             = my_iterator
        ex_element              = my_element.

    WHILE NOT my_element IS INITIAL.

** prop 'swfcn_p_param_optional' is not defined when import flag is not set
      IF NOT my_element->query( properties = swfcn_p_param_import ) = space.

        IF my_element->query( properties = swfcn_p_is_null ) = space AND
          my_element->query( properties = swfcn_p_is_initial ) = space.

        ELSE. "obligatory element is null or initial

          return_code = 999.
          CLEAR: return, wa_return.
          wa_return-message = text-005.
          wa_return-system  = rfc_destination.
          APPEND wa_return TO return.
          RAISE remote_not_supported.

        ENDIF.

      ENDIF.

      my_element = my_iterator->get_next( ).

    ENDWHILE.

  ENDIF.


  IF NOT container_ref IS BOUND.
    return_code = 999.
    CLEAR: return, wa_return.
    wa_return-message = text-008.
    wa_return-system  = rfc_destination.
    APPEND wa_return TO return.
    RAISE internal_error.
  ENDIF.


************* Start workflow ***************
  CLEAR: workitem_id, return_code, return.
  CALL FUNCTION 'SRM_WAPI_START_WORKFLOW'
    EXPORTING
      rfc_destination     = rfc_destination
      sap_release         = sap_release
      task                = task
      input_container_ref = container_ref
    IMPORTING
      workitem_id         = workitem_id
      return_code         = return_code
    TABLES
      return              = return.

  IF NOT return_code IS INITIAL OR workitem_id IS INITIAL.
```

```
    RAISE internal_error.
  ENDIF.

ENDFUNCTION.
```

5 Testing the Scenario

1) Within a record, create an element for the element type created in step 3. The dialog box for entering workflow import parameters appears.
2) Enter the parameters. In the test, use your own user name.
3) Open the Business Workplace (transaction SBWP). You should have received a work item.
4) Double-click the work item. The record should now be displayed.