How to Trigger Workflow from Web Dynpro ABAP – Part 1

Applies to:
This document applies to SAP ECC 4.6 and above. For more information, visit the ABAP homepage.

Summary
This Article helps how to create Business Objects and Workflow.

Author: V Kiran Kumar Reddy
Company: Enteg Infotech
Created on: 02 July 2011

Author Bio
V Kiran Kumar Reddy is working in Enteg Infotech, Bangalore as ABAP/ Web Dynpro ABAP Developer Trainee. He is very much interested in Web Dynpro ABAP and ABAP.
# Table of Contents

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

**Creating Business Object** .......................................................................................... 3
  - Step 1: Create Business Object .............................................................................. 3
  - Step 2: Creating Events .......................................................................................... 5
    - Creating Event Parameters ................................................................................. 6
  - Step 3: Creating Methods ...................................................................................... 8
    - Creating method parameters ............................................................................. 9
  - Step 4: Method Implementation ............................................................................ 12
  - Step 5: Releasing Business Object and its Components ........................................ 14
    - Releasing Object Type ...................................................................................... 14
    - Releasing Object Type Components .................................................................. 15

**Creating Workflow** .................................................................................................... 17
  - Step 1: Creating Workflow .................................................................................... 17
    - Creating Attributes .......................................................................................... 18
  - Step 2: Registering Start Events for Workflow ...................................................... 21
    - Binding Business Object Attributes to Workflow Attributes .............................. 22
    - Activating Start Event for Workflow .................................................................. 24
  - Step 3: Creating Task ............................................................................................ 24
  - Step 4: Creating Send Mail Step ........................................................................... 31

**Related Content** ....................................................................................................... 35

**Disclaimer and Liability Notice** ................................................................................ 36
Introduction
Here I am going to explain a simple scenario to create Business objects and Workflow.

Scenario: Employee enters his ID and click on a button in Web Dynpro ABAP Application which triggers Workflow and send mail with Employee ID and Employee Name to register for the Participation of Event.

After Completion of this Article, You will be able to:
  ✔ Create Business Objects.
  ✔ Create Workflow.

Creating Business Object

Step 1: Create Business Object
Go to SWO1 transaction and Create a Business Object.

Enter attributes: Object type, name, description, program name and application type. Click on Ok
Enter Package name and Save it in a Package.

Note: here you should save the business object in a package else you cannot release it. Releasing Business Object is described in step 5. Business Objects Saved in Local Objects can’t be released and therefore cannot be Transported across boxes (for e.g. DEV to QAS). Also note that a released Business Object means that it is free from syntax errors.
Step 2: Creating Events

Place the cursor on Events Node and then click on create button.

Enter Event name and Description and Click on OK.
Creating Event Parameters

Now we can see the Created Event under Events Node. Place the cursor on the Event and click on Parameters button.

In the next screen click on create.

It will prompt us to create with ABAP Dictionary field reference. Click on NO.
Enter Parameter Name and Description and click on OK.

Now we can see the Parameter which is created. Click on Save and go back to main screen.
**Step 3: Creating Methods**

Place the cursor on methods and click on Create button.

![Create method dialog](change_object_type_zbus593.png)

It will prompt you to create with function module as template. Click on No.

![Create method dialog](change_object_type_zbus593.png)

Enter Method Name and description and under general tab Uncheck Dialog and click on OK.
Creating method parameters

Now we can see the created method. Place the cursor on method and click on Parameters button.
Now in the next screen click on create button.

It will prompt you to create with ABAP Dictionary field reference. Click on No.

Enter Parameter Name, Description. Under Parameter Attributes check import Parameter. Click on OK.
Now Create an Exporting parameter for Employee ID and click on OK. (Here no need of one more exporting parameter for Employee ID, We can use im_empid. For better understanding I created one more Exporting parameter).

Similarly create an exporting parameter for Employee Name and Click on Ok.

Now we can see the parameters which are created. Click on Save and go back to main screen.
Step 4: Method Implementation

Place the cursor on method and click on Program button. Or Click on Check button as below.

It will prompt you to implement the method. Click on Yes so that it will generate the code automatically.
Now we can see the generated code. Now we just need to do little modification.

We need to fetch the Employee Name for the importing parameter `im_empid`. After Slight Modification the code looks like this.

Click on Save and go back to main screen.
Step 5: Releasing Business Object and its Components

Now we can see the Business Object and its methods and Event are in Modeled State. We need to change the status to implemented and then released.

Releasing Object Type

In order to release business object components first we need to release object type.

Place the cursor on business object. Now go to Edit -> Change Release Status -> Object Type -> To Implemented.
Similarly Place the cursor on business object. Now go to Edit -> Change Release Status -> Object Type -> to released.

Releasing Object Type Components
Place the cursor on method. Now go to Edit -> Change Release Status -> Object Type Component -> To Implemented.
Similarly, place the cursor on the method. Now go to Edit -> Change Release Status -> Object Type Component -> to released.

Do the same for Event FIRE.

Now we can see the object type and its Components are in released state (with tick mark beside them).
Place the cursor on business object and click on generate button to generate the Business Object.

Creating Workflow

**Step 1: Creating Workflow**

Go to SWDD Transaction and click on Create button to create new Workflow.
Creating Attributes

Under Navigation Area Work flow container will be selected. If not select Work Flow container as shown below.

Under Workflow Container double click to create as shown below.
Enter Element Name, Description and select the created business object.

Under Properties Tab select import, export and click on Ok.
Now Double click again to create `emp_id` Element which stores the Employee ID.
Enter attribute name, description and enter type as shown below. You can enter the data type directly by selecting ABAP dict. Data type.

Under Properties tab check import, export and click on Ok.
Similarly create one more attribute `emp_name` to store Employee Name.

Under Properties tab check import, export and click on Ok.

Click on Save and Save the Work flow.

**Step 2: Registering Start Events for Workflow**

Click on basic data (hat symbol) as shown below.
Go to Start events Tab and select Category as BO (Business Object), Enter Business Object Name and Enter Event Name (you can select event from F4 help) as shown below.

Binding Business Object Attributes to Workflow Attributes

Click on binding as shown below

Drag and drop **empid** from Event container to **emp_id** of Work Flow Container to map **empid** attribute of Business Object to **emp_id** of Work Flow Container.
Now you can see empid of Business Object is mapped with emp_id of Workflow Container. Click on OK.

Check the below process flow to understand the binding concept.

We will send the empid to Business Object from Web Dynpro ABAP

From Web Dynpro ABAP we will pass the Employee ID to Business Object Container. Now by binding we will get the Employee ID from Business Object Container to Workflow Container.
Activating Start Event for Workflow

Click on Activate as shown below.

Once activated it turns to green.

This means once the FIRE event of Business Object is triggered the Workflow will Start.

Click on Save and go back to main screen.

Step 3: Creating Task

Double Click on the Undefined Step as shown below.
Select Activity Step and click on Ok.

Click on Create Task as shown below.

Enter abbreviation, Name. Under Object method, select Business Object. Enter Business Object Name and Method Name (You can select method from F4 help) as shown below.

Under Execution Select Background Processing, since no user interaction is required for our scenario.
Click on Save and Click on Back Button.

It will prompt to Transfer missing elements. Click on Yes. (You can also click on No and create elements manually).

Now Click on Ok.
Click on binding as shown below.

You can see the binding automatically exists with the matching types, since we click on yes in the above step.

Here im_empid of workflow Container to im_empid of Task Container. But we have to bind emp_id of Workflow Container to im_empid of Task Container.

Similarly we have to bind ex_empid of Task Container to emp_id of Workflow Container and ex_empname of Task Container to emp_name of Workflow Container.
Select the Line and Click on delete icon to delete the existing mapping as shown above.

Now drag and drop `emp_id` from Workflow Container to `im_empid` of task container as shown below.
Similarly drag and drop ex_empid and ex_empname from Task Container to Workflow Container as shown below.

Now we can see the Required Mapping. Click on Ok.
Check the below process flow to understand the binding concept.

We mapped emp_id of Workflow Container to im_empid of Task Container. It means we are passing Employee ID from Workflow Container to Task Container, which calls SEND_VALUE method of Business Object, in which we are fetching Employee Name and exporting Employee ID and Employee Name.

Now from Task Container we are passing ex_empid to emp_id of Workflow Container and ex_empname to emp_name of Workflow Container.

Now Click on Transfer Button.
Step 4: Creating Send Mail Step

Now we can see the Created Task. Click on Create as shown below to create the Mail Step.

Select Send Mail Step and Click on Ok as shown below.
Enter Subject and Click on insert expression to insert Value as shown below.

Select EMP_ID (which holds Employee ID) and click on Ok.
Enter the Body of Mail and click on insert expression under body to insert Value.

Enter body and click on insert expression to insert value

Now Click on transfer button.

Here I selected expression and WF initiator, which sends mail to initiator of workflow. We can also send to particular user, or to external mail.
Enter abbreviation and Name and click on Ok.

Now the Work Flow looks like this.

Now Our Workflow is ready.

In the next part How to Trigger Workflow from Web Dynpro ABAP - Part 2 we will see how to create Web Dynpro ABAP Application and Trigger Workflow from Web Dynpro ABAP.
Related Content

How to Create Business Objects
Workflow Programming - SAP Help
Web Dynpro ABAP
Integration of Workflow with Web Dynpro ABAP
Web Dynpro ABAP homepage

For more information, visit the ABAP 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.