This Blog is a Guide to Send Your Web Dynpro Java or Web Dynpro ABAP applications to Universal Work List as Tasks. It is needed because the Module Pool Applications behave abnormally in UWL and problems in using Decision screens of workflow in UWL.

**Target Audience:** Workflow Gurus and Web Dynpro Java/ABAP Developers.

**Basic Knowledge Required**
- Knowledge on Workflows (Creating Workflow Templates and Tasks)
- Basic Knowledge on Web Dynpro Java (to Integrate UWL with Web Dynpro Java) - OR -
- Basic Knowledge on Web Dynpro ABAP (to Integrate UWL with Web Dynpro ABAP)

**Prerequisites**
Your Portal UWL should be configured first to fetch Tasks from Back-end R/3.

**Create a Simple Workflow Template**
1. Start Transaction `pftc` to create a workflow Template.
2. In the Workflow Builder create an Activity Task and give any Dummy Business Object and method (This Business Object has no relevance here, We want Our Web Dynpro Application to be executed instead the BOR Method)
3. Add a Send Mail Task after the above Activity Task, which sends mail to the Initiator.
4. You can Assign Agents for the Workflow and the Tasks or you can make it General.
Create a Web Dynpro Java Application to Trigger Workflow

1. Create a Simple Web Dynpro Java Development Component to trigger the workflow
2. Import Model `SAP_WAPI_START_WORKFLOW', this is the RFC Function Module used to trigger workflow.
3. `SAP_WAPI_START_WORKFLOW' RFC FM takes TASK as an Import Parameter which is our Workflow Template. E.g. WS90000024. We can pass the AGENT name and fill the Workflow Container which is passed as TABLES parameter to this FM.
4. Build and Deploy the Application.

Code to Trigger Workflow:

```java
Sap_Wapi_Start_Workflow_Input();
wdContext.nodeSap_Wapi_Start_Workflow_Input().bind(startWFInput);
// set Task
startWFInput.setTask('WS90000024');
// set Agent
Swragen agents = new Swragen();
agents.setOtype("US");
agents.setObjid("USERNAME");
startWFInput.addAgents(agents);

try {
    wdContext.currentSap_Wapi_Start_WorkflowInputElement().modelObject().execute();
} catch (WDDynamicRFCExecuteException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
When we run this Application, the Workflow Starts and a Task should be waiting in UWL, when user clicks the task link in UWL a Web Dynpro Java Application is to be opened and inside the application we handle the completion logic of the Task.

**NOTE:**
When the Workflow is started, the workflow engine assigns a Unique ID to each Task involved in the workflow process, this is called as Work Item ID. Upon Clicking the Task Link in UWL, UWL sends this Work Item ID to the Web Dynpro Application in the Request Object.

**Create a Web Dynpro Java Application for the Task in UWL**
1. Create a new Web Dynpro Development Component, which is called when the task is clicked in UWL
2. Read the Work Item ID from the Request Object and Store it in Context or any Variable.

   ```java
   String workItemID = WDProtocolAdapter.getProtocolAdapter()
                      .getRequestObject().getParameter("WI_ID");
   ```
3. We use 'SAP_WAPI_WORKITEM_COMPLETE' RFC to Complete the Task Import it as a model, and execute it passing the Work Item ID as import Parameter
4. Build and Deploy Your Application in the Portal

We can use Other RFC Function Modules to Change and Read the Task Container Elements and Add Attachments to the Task etc..
- Read Task Container Elements - SAP_WAPI_READ_CONTAINER
- Change Task Container - SAP_WAPI_WRITE_CONTAINER
- Add Attachment - SAP_WAPI_ATTACHMENT_ADD

**Code to Read Work Item ID and Complete the Task:**

```java
String workItemID = WDProtocolAdapter.getProtocolAdapter()
                    .getRequestObject().getParameter("WI_ID");
wdContext.nodeSap_Wapi_Workitem_Complete_Input().bind(
    new Sap_Wapi_Workitem_Complete_Input());
wdContext.currentSap_Wapi_Workitem_Complete_InputElement()
    .setWorkitem_Id(workItemID);
try {
   wdContext.currentSap_Wapi_Workitem_Complete_InputElement()
        .modelObject().execute();
} catch (WDDynamicRFCExecuteException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
}
```
Configure Your Workflow Task with Your Web Dynpro Java Application:

1. Start Transaction SWFVISU, Choose New Entries
2. Enter your Task ID E.g. TS90000013 and Select Web Dynpro Java Application

3. Enter Visualization Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION</td>
<td>Application Name E.g. TestApp</td>
</tr>
<tr>
<td>PACKAGE</td>
<td>Vendor name/development Component Name E.g. sap.com/test</td>
</tr>
<tr>
<td>DYNPARAM</td>
<td>WI_ID=${item.externalId}</td>
</tr>
<tr>
<td>SYSTEM_ALIAS</td>
<td>Alias name of the system where the application is deployed. If it is the Portal where the Web Dynpro Application Deployed, the Alias is SAP_LocalSystem.</td>
</tr>
</tbody>
</table>

4. Login to Enterprise Portal as Administrator and Re-Register Your Back-end System
Procedure for Web Dynpro ABAP Applications to Read Work Item ID:

As I have mentioned earlier, UWL passes the Work Item ID to the Application in Request Object. In Web Dynpro ABAP we have another way of retrieving this Work Item ID.

2. Go to the Main Window, switch to the tab Inbound Plugs, double click on the Default Plug, which will take you to the HANDLEDEFAULT method.
3. Add a Parameter to this method called WI_ID of Type String.
4. The above method is called when the Application is called and the WI_ID is passed to it as parameter.
5. Get the Value of WI_ID in the method and store it in component controller so as to use it in views to complete the work item using SAP_WAPI_WORKITEM_COMPLETE, Read the Container using SAP_WAPI_READ_CONTAINER.
NOTE: All the Function Modules which are mentioned for Web Dynpro Java can be used in Web Dynpro ABAP; we can directly call any function module with fewer efforts unlike Web Dynpro Java. Take the full advantage of being in the same ABAP stack.

**Configure Your Workflow Task with Your Web Dynpro ABAP Application:**

This configuration same for Both Web Dynpro Java and Web Dynpro ABAP, Instead choosing Web Dynpro Java Application, choose Web Dynpro ABAP.

Change the Visualization Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION</td>
<td>Application Name E.g. TestApp</td>
</tr>
<tr>
<td>NAMESPACE</td>
<td>Always sap (All WD Components are under sap Namespace)</td>
</tr>
<tr>
<td>DYNPARAM</td>
<td>WI_ID=${item.externalId}</td>
</tr>
<tr>
<td>SYSTEM_ALIAS</td>
<td>Alias name of the system where the application is deployed</td>
</tr>
</tbody>
</table>

Web Dynpro ABAP Application run in R/3 System, give the name of the System Alias of the R/3 System.