Developing a Universal Worklist Connector for YAWL (Yet Another Workflow Language)

Applies to:
SAP NetWeaver 2004 SPS 18, SAP NetWeaver 2004s SPS 9

Summary
Since the release of the UWL API at SAP TechEd 2006 the Universal Worklist (UWL) allows the integration of non-SAP systems using the Java API. This article demonstrates how easy it is to connect a third-party system using the API. To demonstrate the straightforward implementation approach, the article shows how to develop a UWL connector for an open source workflow engine (YAWL). YAWL is the reference implementation of a workflow engine based on workflow patterns and Petri net (also known as a place/transition net or P/T net), and is being used a lot in research.

Thanks to Prof. Wil van der Aalst for encouraging the use of YAWL to demonstrate this integration and to Lachlan Aldred for his support and feedback. Thanks also to the researchers at Eindhoven University of Technology and the Queensland University of Technology who have developed YAWL. Finally, thanks to SAP Research in Brisbane who are experts in Business Process Management and have a long standing work relationship with University of Queensland (UQ) and Queensland University of Technology (QUT).

Author: Lars Rüter
Company: Regional Implementation Group (RIG), SAP Australia Pty. Ltd.
Created on: 14 June 2006

Author Bio
Mr. Lars Rüter is a Senior SAP NetWeaver Consultant who works at SAP Australia Pty. Ltd. for the Regional Implementation Group (RIG) Asia Pacific. He has extensive experience in Java development, SAP NetWeaver Master Data Management and SAP NetWeaver Portal. Mr. Rüter is also a co-author of the SAP Education training course “Knowledge Management and Collaboration.”
Table of Contents

Introduction ...................................................................................................................................... 2
Setting Up the IDE ........................................................................................................................... 4
Configuring the YAWL Connector ................................................................................................... 8
  Creating a Portal System ............................................................................................................. 8
  User Mapping ............................................................................................................................. 10
  UWL Configuration ..................................................................................................................... 11
  Testing the Connection .............................................................................................................. 11
Creating a Portal System ............................................................................................................. 8
User Mapping ............................................................................................................................. 10
UWL Configuration ..................................................................................................................... 11
Testing the Connection .............................................................................................................. 11
Coding in Detail ............................................................................................................................. 11
  External Libraries ....................................................................................................................... 11
  YAWL Provider Connector Implementation ............................................................................... 12
Limitations ...................................................................................................................................... 14
Disclaimer & Liability Notice .......................................................................................................... 14
Related Content ............................................................................................................................. 14
Copyright ........................................................................................................................................ 15

Introduction

The Universal Worklist (UWL) enables SAP NetWeaver Portal users to manage their work by bringing together notification items from different systems. The UWL delegates the task of connecting to a backend system and retrieving items to a so called UWL Connector. The following UWL Connectors are delivered together with the SAP NetWeaver Portal:

- WebFlow Connector – fetches work items from SAP Business Workflow.
- AdHocWorkflow Connector – fetches collaboration tasks from SAP AdHoc Workflow.
- Alert Connector- fetches Alerts and follow-up activities from SAP Alert Management System.
- ActionInbox Connector- fetches document related notifications from SAP Knowledge Management.

Generic ABAP Connector – For ABAP based providers, UWL simplifies the connector API by doing much of the plumbing work in this Generic ABAP connector. The ABAP based provider needs to implement the interface IF_UWL_ITEM_PROVIDER to supply items to UWL.

Work items from other systems can be displayed in the UWL as well, provided you have an appropriate connector that knows how to retrieve these items from the respective system. The UWL provides a Java API for the development of such a connector. This document describes how you can develop a connector for the open source workflow engine YAWL.

YAWL is a workflow language built upon two main concepts: workflow patterns and Petri net. It was developed by taking Petri nets as a starting point and adding mechanisms to allow for a more direct and intuitive support of the workflow patterns identified. YAWL supports the control-flow perspective and the data perspective. YAWL is open source and distributed under the GNU Lesser General Public License.

Some claims about YAWL:

- OR, AND, and XOR splits/joins
- Single tasks are atomic ones or sub-workflows
- Web Services may be used as atomic tasks of a workflow
- Explicit parallelism (multiple instances of an atomic task)
- Cancellation of a part of a workflow by a particular task

Below you can see screenshots of the solution presented in this article. The first figure shows the checked out work items in the YAWL web interface. In the figure below the same work items are displayed in the UWL through the SAP NetWeaver Portal.

Figure 1: List checked out work items with YAWL web front end

When displaying the YAWL work items in the UWL you can choose different filters depending on which work items you want to display. You can show active (checked out), available or all work items.
Setting Up the IDE

In case you don’t want to start a project from scratch and decide to use or enhance the existing YAWL connector this section explains how to set up the exiting project in the NWDS.

Download and extract the source code archive YAWL_Connector_SRC.zip to your development configuration root folder under [dev conf root]\dtc\LocalDevelopment\DCs. Remove the read-only attributes from all files below the sap.com folder.
Open your SAP NetWeaver Developer Studio (NWDS) and switch to the Development Configurations perspective. Under Local DC’s you should now see at least the two DC’s uwl.yawl and uwl.yawl.lib. Select and right click the uwl.yawl.lib DC. Select create project. Repeat the same steps for the uwl.yawl.

You can now switch to the Package Explorer perspective to view the local DC’s that you have imported. Open the projects if they are not already open.
At this stage you may notice some error messages which are a result of the project not being built. The next step is therefore building the project. To build the project, follow the steps shown in figures 6 and 7.
You have successfully imported and build the YAWL connector. The final step in this section is to deploy the connector to a SAP NetWeaver Portal. Start the deployment as shown in the figure below. After deployment has finished follow the instructions in the next section to configure the connector for use in the portal.

Figure 8: Deploy YAWL connector
Configuring the YAWL Connector

After you have deployed the YAWL connector to a SAP NetWeaver Portal some additional configuration steps are required before you can see some YAWL work items in the UWL. At this stage it is assumed that you have already installed YAWL either with or without the PostgreSQL database. You should also have created some work items waiting to be checked out or edited. Please consult the YAWL documentation for instructions on how to do this.

Creating a Portal System

The YAWL connector needs to know where in your system landscape your YAWL server is located. The YAWL connector retrieves this information from the portal system landscape where it has to be configured before the connector can be used.

The location of a YAWL system is described by a URL that contains the hostname and port of the installation. Here is an example: http://localhost:8080/yawl/lib. The URL is always the same except for the hostname and port that need to be changed according to your installation.

The figure below shows how to create a new Web Service System that you need for the YAWL connector.

![Figure 9: Create a new Web Service System](image)
Under property category select Web Service as shown below. Fill in the WSDL URL in the appropriate input field.

![Figure 10: Setting the URL of the YAWL system](image)

Also select the property category User Management to maintain the logon method and user mapping type as shown in figure 11.

![Figure 11: Configuring User Management Setting](image)

In order to reference the portal system from the UWL configuration and also to enable user mapping you need to create an alias for the system. Figure 12 provides an example.
Figure 12: Create a system alias

**User Mapping**

The YAWL connector does require a user and password to connect and authenticate to the YAWL engine. Therefore you must do a user mapping before you can retrieve work items from the YAWL engine. This can be done using the personalize page.

Figure 13: Click on personalize to do a user mapping
UWL Configuration

The last configuration step requires you to create a new UWL system. This can be done in the Universal Worklist configuration. You only need to select the “YawlConnectorProvider” from the drop down box and enter the system alias that you have created together with the portal system in the first step.

Testing the Connection

You can now open the Universal Worklist and should see your YAWL work items in the task list. If you cannot see any work items check the configuration steps above again. Make sure you have mapped a valid user. A default user that you can use for testing the connection is “admin” with the password “YAWL”. Also check that you actually have work items using the YAWL native web interface.

Coding in Detail

External Libraries

All libraries required for this connector implementation which are not referenced by a standard SAP DC are put into an external library DC project. The External Library DC will contain all Java libraries which are not yet defined in SAPs standard DC delivery. This mainly contains the UWL and portal libraries, but also any third party library which should be used within the connector implementation. Refer to figure 15 for a detailed list of required libraries.
It is recommended to use the UWL connector wizard for NWDS when implementing a new connector. If you have used the UWL connector wizard just follow the @TODO's provided in the source code template. The main job is to implement the `getItems()` method.

First we look at how we can retrieve work items from YAWL. The YAWL interface is currently based around the REST architectural style instead of SOAP. Communication with the YAWL engine is done by using the appropriate interfaces from the Interface B library provided as part of the YAWL package. The interface relevant for you to retrieve work items is called `InterfaceB_EnvironmentBasedClient`. It provides a Java API for remote applications to effectively communicate with the YAWL engine.

```java
InterfaceB_EnvironmentBasedClient interfaceBClient = new InterfaceB_EnvironmentBasedClient(yawlSystemEndpoint);
```

The line above creates an object the represents the interface for remote access to YAWL. Before you can perform any operations on the interface you must authenticate and obtain a session ticket that can be used for subsequent requests.

```java
String sessionHandle = this.interfaceBClient.connect(m_user, m_pwd);
```

Once you have the session ticket you can retrieve work items by calling the appropriate method on the `interfaceBClient` object.

```java
List workItems = interfaceBClient.getCompleteListOfLiveWorkItems(sessionHandle);
```

Work items are returned as a list of `WorkItemRecord` objects. When you iterate through this list each work item needs to be converted into an UWL item. This is done by coping attributes from the `WorkItemRecord` object to an `Item` object that represents an UWL item. This is not done in the `getItems` method but in a separate private method `mapYawlToUwlItem`.

The following source code lists the `getItems` method in the `YawlConnectorProvider` class. The important steps are highlighted in bold.

```java
public ConnectorResult getItems(UWLContext context, String itemType, ConnectorFilter connectorFilter, String system) throws ConnectorException {
    // @TODO 4: Implement getItems
    ConnectorResult result = null;
    List retItems = new ArrayList();
    if (this.interfaceBClient == null) {
        String yawlSystemEndpoint = this.getYawlSystemEndpoint(system);
        this.interfaceBClient = ...
        String sessionHandle = this.interfaceBClient.connect(m_user, m_pwd);
    }
    List workItems = interfaceBClient.getCompleteListOfLiveWorkItems(sessionHandle);
    ...
new InterfaceB_EnvironmentBasedClient(yawlSystemEndpoint);

String sessionHandle;
try {
    IUserMappingData iumd = this.getMapping(context.getUser(), system);
    String m_user = this.getMappedUser(iumd);
    String m_pwd = this.getMappedPassword(iumd);
    sessionHandle = this.interfaceBClient.connect(m_user, m_pwd);
    loc.infoT("Authentication successful. Got YAWL session handle: "+
             sessionHandle);

    List workItems =
                  interfaceBClient.getCompleteListOfLiveWorkItems(sessionHandle);
    Iterator iter = workItems.iterator();
    while(iter.hasNext()) {
        WorkItemRecord yawlItem = (WorkItemRecord) iter.next();
         Item uwlItem = mapYawlToUwlItem(yawlItem, system,
                                  context.getUser(), m_user, sessionHandle);
         retItems.add(uwlItem);
    }
} catch (IOException e) {
    throw new ConnectorException(e);
} catch (JDOMException e) {
    throw new ConnectorException(e);
}

ProviderStatus status =
    new ProviderStatus(true,system, YawlConnectorProvider.CONNECTOR_ID);
//@TODO 5: Decide if DeltaResult or Snapshot result should be created.
result = ConnectorResult.createSnapshotResult(
    new ItemCollection(retItems),status);
return result;

You will notice that the method getCompleteListOfLiveWorkItems returns a list of all work items in the YAWL engine independent of the user or state. For the end user it is beneficial to see available (not checked-out) and checked-out work items in separate views in the UWL. Also the user should only see her own checked-out work items.

To get different views in the UWL for checked-out and available work items you need to define separate items types in the YawlConnectorProvider class.

private static final String[] uwlItemTypes = {
//@TODO 1: Add item types supported by the connector here.
    "uwl.task.yawl",
    "uwl.task.yawl.active", // checked-out
    "uwl.task.yawl.available", // not checked-out, but available for check out
};

Next you apply a filter that assigns the appropriate item type when creating an UWL item. This is best done in the mapYawlToUwlItem method.

String uwlItemType = uwlItemTypes[0]; // default item type
if (yawlItem.getStatus().equals(WorkItemRecord.statusDeadlocked) ||
     (yawlItem.getStatus().equals(WorkItemRecord.statusExecuting) &&
      (yawlItem.getWhoStartedMe().equals(mappedUser))))) {
Developing a Universal Worklist Connector for YAWL (Yet Another Workflow Language)

```java
uwlItemType = uwlItemTypes[1]; // Active Work
}
if (yawlItem.getStatus().equals(WorkItemRecord.statusEnabled) ||
    yawlItem.getStatus().equals(WorkItemRecord.statusFired)) {
    uwlItemType = uwlItemTypes[2]; // Available Work
}
```

Finally you create the new item types and views in the UWL configuration file located in your project folder under `\dist\PORTAL-INF\classes\uwl_yawl.xml`. If you modify the configuration file and deploy your project a restart of the J2EE engine is required (tested with NW04s SPS 7) before the changes will take effect.

**Limitations**

The current implementation of the YAWL connector is intended to get you started and to show how easy it can be to implement a connector for a third party system. The implementation is by no means complete. Below is a list of features that you should implement.

- **Time and date conversion**
  The YAWL API’s return a timestamp in a specific format (for example Jun:09 14:45:33). A `java.util.Date` object is required to create a UWL item. Currently the YAWL connector only uses the current date and time when creating a UWL item. Instead you should parse the timestamp returned from the YAWL API and use it to create a `java.util.Date` object. This can then be used when creating the UWL item.

- **Action handling**
  No action handling is implemented for this connector at the moment. You can not check out or edit work items through the UWL. Both actions are simple to implement. To check out a YAWL work item you can use the Interface B YAWL API. To edit a checked out work item you use the `setExecutionUrl()` on the UWL item to assign a URL that opens the YAWL web interface. Alternatively you could develop your own WebDynpro screens for specific edit functionality.

**Disclaimer & Liability Notice**

You have a royalty-free right to use, to modify, to reproduce, and to distribute the YAWL connector sample in any way you find useful. This is provided that you agree that SAP has no warranty, no obligations, and no liability for its use or for the information provided within. The connector and source code is provided free of charge for educational purposes only and is considered a sample. If you want to use the connector, or any part thereof, in a production solution, you are responsible for reviewing, for modifying, for testing, and for supporting the component yourself.

Warning: The YAWL connector and associated sample code is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and/or fitness for a particular purpose. Use at your own risk.

**Related Content**

- YAWL: Yet Another Workflow Language
- YAWL API: Interface B description for Beta 7 in PDF format
- SDN Blog by Alan Rickayzen: Opening up the portal universal worklist for third party software
- SDN Blog by Alan Rickayzen: Gap plugged - Notifications in the universal worklist
- Interesting example of using (extended) YAWL language to speed-up traditional personnel-based process
- REST: Representational State Transfer
Copyright

© Copyright 2006 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.