

Incorporating Workflow Into Sun Portals With Sun Java Composite Application Platform Suite

By Vihang Pathak and Marina Sum
March 2, 2007



THE NETWORK IS THE COMPUTER™

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Copyright (C) 2007 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd. X/Open is a registered trademark of X/Open Company, Ltd.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc.

Sun, Sun Microsystems, the Sun logo, Solaris, Sun Studio, OpenSolaris, NetBeans, Java, Java EE, Java SE, Sun Java Enterprise System, and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

This product is covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Contents

A Macroscopic View	2
Interfaces for Managing Tasks	2
Overview of Integration	3
Example	3
Fulfilling the Prerequisites	4
Creating a Portlet for Input	4
Creating a Portlet for Output	5
Creating and Configuring a Business Process	6
Creating a Business Process	6
Mapping Data	7
Assigning Tasks	8
Creating an Environment	11
Configuring the Environment	12
Building and Deploying the Project	15
Creating a Portlet for Managing Tasks	17
Understanding the Execution Process	17
Taking the Steps	18
Browsing the Sample Code	19
Conclusion	20
References	20

Figures

Figure 1	Example of a Combination of Automated and Manual Workflow in a Portal2	
Figure 2	Process of Integrating Portal Server With Java CAPS for Human Workflow	3
Figure 3	Portlet for Input	.5
Figure 4	Portlet for Output	.5
Figure 5	Creation of Business Process	.6
Figure 6	Mapping of Input	.7
Figure 7	Mapping of Output	.8
Figure 8	Access of Edit Task Assignment Pane	.9
Figure 9	Assignment of Users for Tasks	.10
Figure 10	LDAP Connection Properties	.10
Figure 11	Creation of Logical Host	.11
Figure 12	Creation of Integration Server	.12
Figure 13	Properties for WML Connector Database	.13
Figure 14	Location for Output File	.14
Figure 15	Generation of Connectivity Map	.14
Figure 16	Properties for the Input File	.15
Figure 17	Creation of Deployment Profile	.16
Figure 18	Task Management in Action by Portlet	.18
Figure 18	Task Management in Action by Portlet	.22

Incorporating Workflow Into Sun Portals With Sun Java Composite Application Platform Suite

Traditionally, performing tasks on multiple portals is tedious and inefficient as enterprises adopt back-end business processes and must then continually manage them. Hence, demand has steadily mounted for the capability to perform tasks on just the portal itself instead of on multiple sites with a single-point solution; that is, a solution that not only handles the process interactions but also enables end users to seamlessly perform tasks as “hooks” in the process.

Bear in mind that automation of business integrations requires human steps on portals, often accomplished through proprietary applications. By integrating human workflow into portals, you ensure that all the tasks, whether performed by humans or by software, occur in one place. Toward that end, an integration of [Sun Java System Portal Server](#) (henceforth, Portal Server) and the [Sun Java Composite Application Platform Suite](#) (Java CAPS) delivers an effective solution.

Portal Server, part of [Solaris Enterprise System](#), enables efficient, identity-based, and cost-effective deployment of enterprise portals. Among Portal Server’s many robust capabilities are services for building community portals and tools for content development. Portal Server is **free** for [download](#).

With Java CAPS, enterprises can build and deploy a service-oriented architecture (SOA) platform for integrating and reusing applications and for developing Web services. Included in Java CAPS are the Java Enterprise Service Bus (ESB) Suite and Java Business-to-Business (B2B) Suite. Java CAPS will [soon be available for free download](#).

Through an example, this article describes how to automate human workflow by integrating Portal Server with Java CAPS. It is assumed that you are familiar with Portal Server and Java CAPS components, such as Enterprise Designer, eInsight, eVision, eWays, and Worklist Manager (WLM). For details, see the related documentation.

A Macroscopic View

Consider the scenario of a chain of tasks, some automated and others manual. Subsequent to automated processing, manual tasks and then more automated processing follow. Managing the tasks and processing for optimum efficiency is, obviously, an important aspect of that scenario.

Figure 1 illustrates an example.

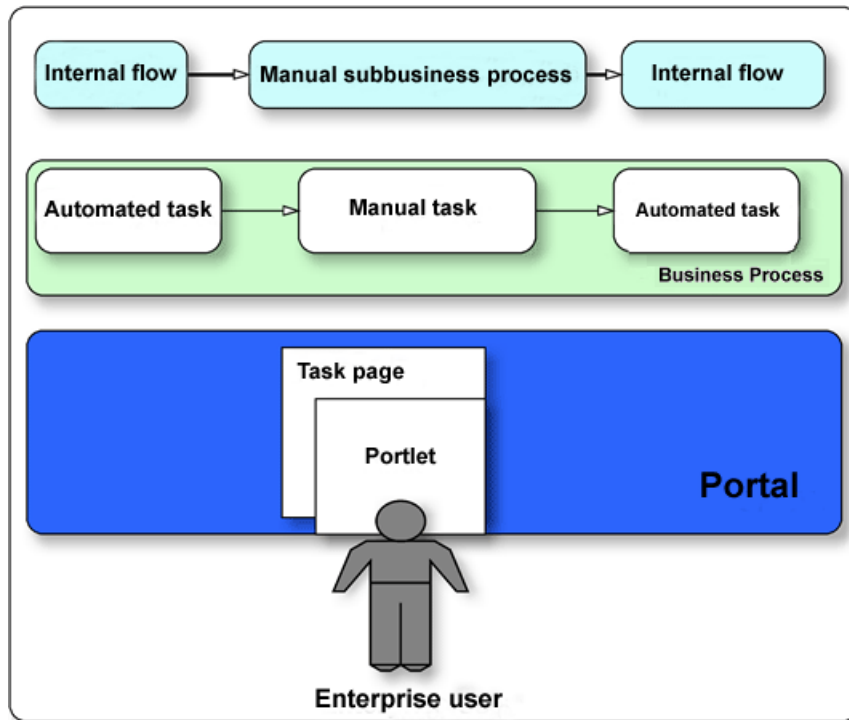


Figure 1 Example of a Combination of Automated and Manual Workflow in a Portal

Interfaces for Managing Tasks

In Java CAPS, you can define both the business processes that might require user interactions and the user interfaces (UI) for those interactions. However, you cannot customize the UI for managing tasks.

The integration process described in this article eliminates that constraint. By creating portlets with the WLM API for submitting and retrieving tasks, also for obtaining task status, you can, in fact, build a UI for managing tasks. The usual sequence of events looks like this:



Overview of Integration

Figure 2 illustrates an example of the process of integrating Portal Server with Java CAPS for human workflow.

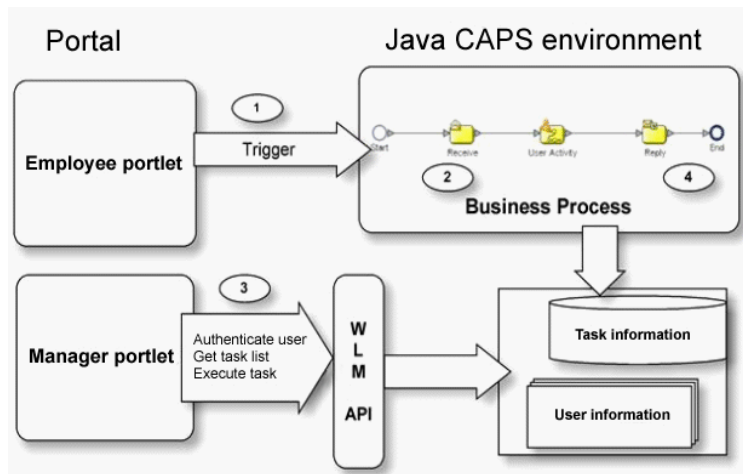


Figure 2 Process of Integrating Portal Server With Java CAPS for Human Workflow

Example

This section walks you through the example, which is a simple five-step process:

1. An employee requests a leave of absence on the portal, triggering a Business Process on the back-end Java CAPS. This step initiates a task that requires human interaction.
2. A manager portlet displays the task, pending interaction with the user—in this case the employee’s manager—whom the portlet helps manage the subsequent tasks.
3. Enabled by the manager portlet, the manager approves or rejects the request, an action that signals the Business Process to proceed.
4. The Business Process writes the output to a file.
In real life, subsequent to approval or rejection by the employee’s manager, the Business Process can pass the response from the manager portlet to other actions and then write the final output to a destination.
5. A display output portlet displays the response on the portal.

The preceding process clearly demonstrates that task management and integration of business processes are two key complementary Java CAPS components for Portal Server.

Note: A screencast demo will be available soon, at which time this article will point to it.

Fulfilling the Prerequisites

Before digging into the example, perform the following tasks:

- Install Portal Server.
- Install Java CAPS.
- Install a relational database management system (RDBMS).
- Install and configure the Lightweight Directory Access Protocol (LDAP).
- Ensure that both Portal Server and Java CAPS can access each other over the network and that they share a common file storage.

Creating a Portlet for Input

First, create a simple portlet (see Figure 3) to accept and store user input in a location, such as a file in a directory that is accessible to a Java CAPS Business Process, described later in this article. For the procedure on creating portlets, see the article [Developing Portlets with the NetBeans Portlet Plugin](#).

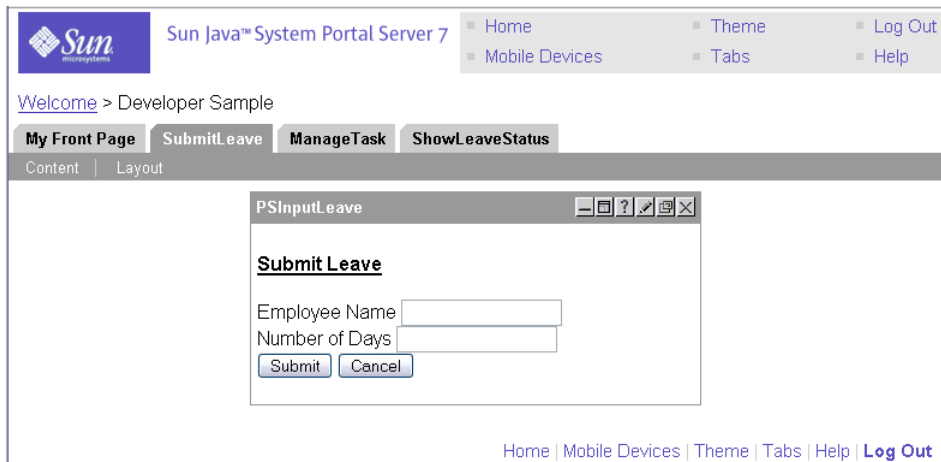


Figure 3 Portlet for Input

Creating a Portlet for Output

Next, create a venue, such as a file or database, for the Business Process output. For simplicity, just create a file. Alternatively, output to a Java Message Service (JMS) receiver in the process to serve multiple users.

Afterward, create a portlet to extract the data from the file and show the data to the end users. See Figure 4.

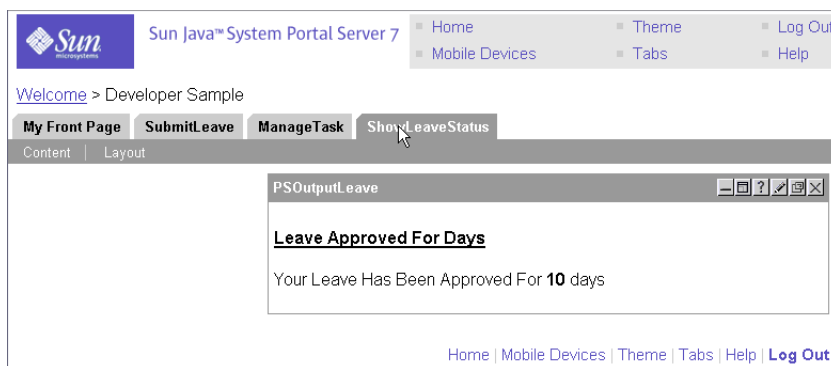


Figure 4 Portlet for Output

Creating and Configuring a Business Process

You are now ready to create and configure a Business Process.

Creating a Business Process

To create a Business Process:

1. Right-click MyRepository in the Sun SeeBeyond Enterprise Designer, choose Create New Project to create a new project, and add a Business Process in Java CAPS.
2. On the left pane, click eWays under Sun SeeBeyond and then File, FileClient, and receive.
3. Drag and drop write to the right pane to create a Business Process.
4. Drag and drop a User Activity API node from the top bar in the right pane onto the Business Process.
5. Connect Start, FileClient receive, User Activity API, File Client write, and End. See Figure 5.

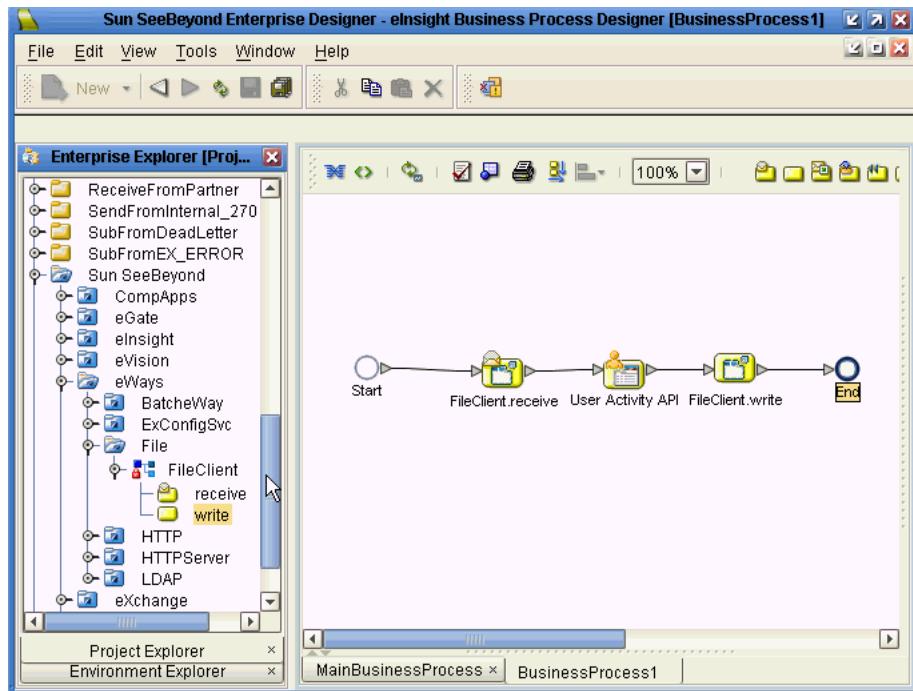


Figure 5 Creation of Business Process

Mapping Data

Now define the rules for data mapping so that data can correctly enter and exit the process. The User Activity API contains task fields that map to the database of the back-end WLM. Hence, map the appropriate input to the `input_data` field in the Task Database by defining the Business Rules in Java CAPS, as follows:

1. Right-click the link between `FileClient.receive` and User Activity API and choose Add Business Rule from the context menu.
2. In the Output pane, connect the label “text” under `FileClient.receive`. Output to the “input” label under WLM Task Input, as shown in Figure 6.
Java CAPS treats the file input as the input in the WLM database.

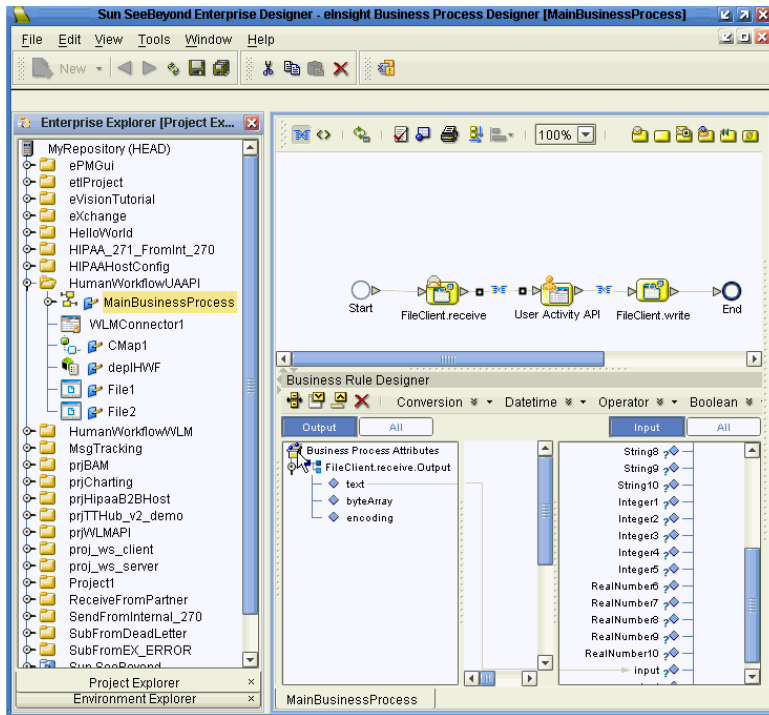


Figure 6 Mapping of Input

3. Similarly, right-click the link between User Activity API and `FileClient.write` and choose Add Business Rule from the context menu. Afterward, connect the WLM “output” label in the Output pane to the “text” label in the Input pane, as shown in Figure 7.

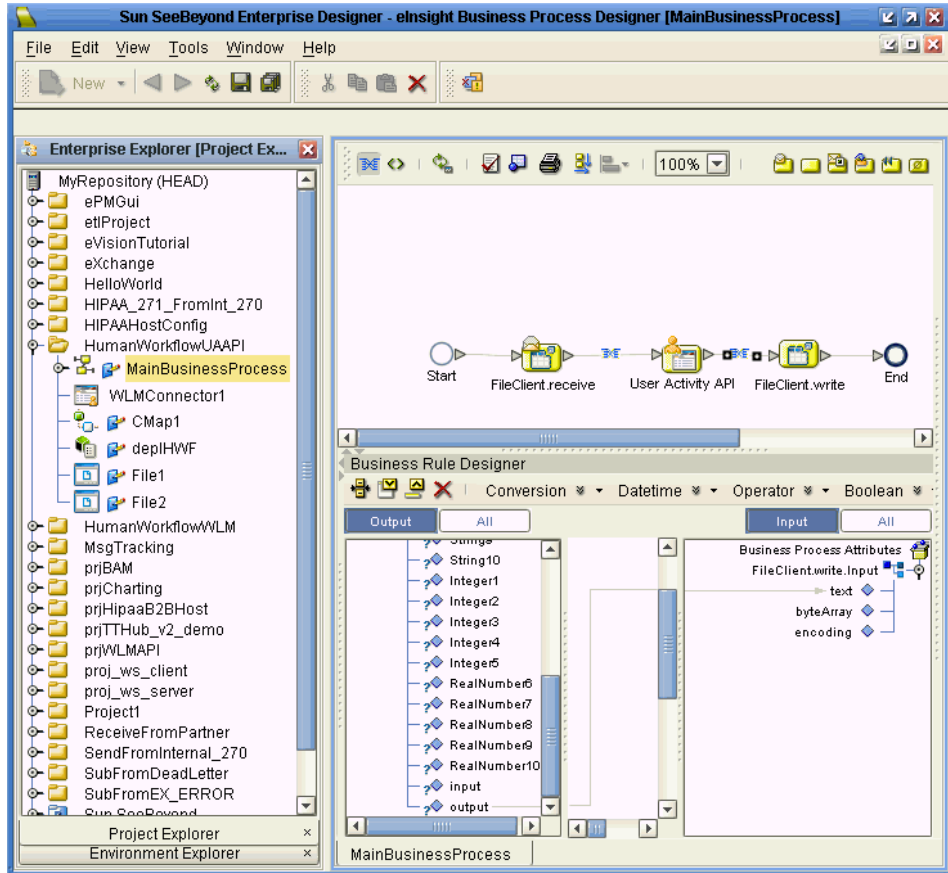


Figure 7 Mapping of Output

Assigning Tasks

Next, specify the users who are authorized to perform the tasks. The user data resides in [Sun Java System Directory Server](#), [Open LDAP Server](#), or [Microsoft Active Directory Server](#). You can dynamically specify the authorized users by defining the rules for the user assignments.

Follow these steps:

1. Right-click User Activity API and choose Edit Task Assignment pane. See Figure 8.

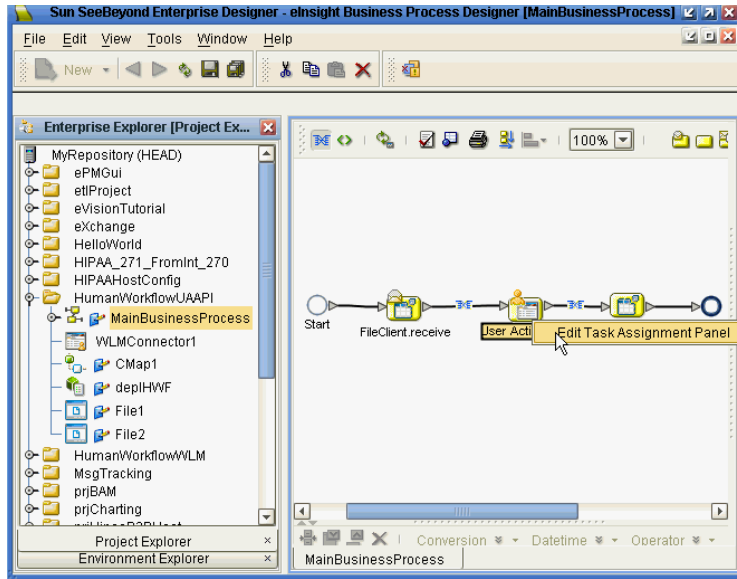


Figure 8 Access of Edit Task Assignment Pane

The Edit Task Assignment pane, in which you define the rules for group assignments, timeouts, and email notifications, is displayed. See Figure 9.

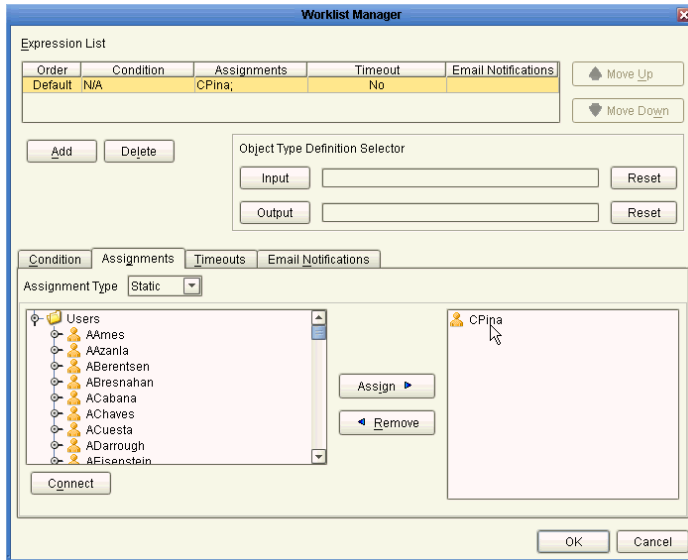


Figure 9 Assignment of Users for Tasks

2. Click Connect near the bottom left.

The dialog box for LDAP connection properties is displayed. Shown is the information that pertains to the host attributes. See Figure 10.

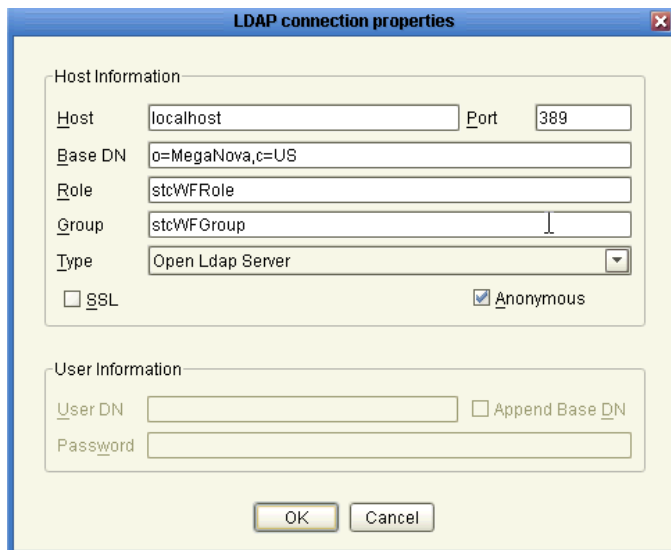


Figure 10 LDAP Connection Properties

3. Assign a user by posting the relevant information from LDAP: Fill in the fields under User Information with the user-specific LDAP credentials.

Note: You can open the relevant database from the Type pull-down menu.

4. Click OK.

Creating an Environment

Next, create an environment, including an Integration Server:

1. Click Environment Explorer on the left pane, right-click My Repository, and choose New Environment from the context menu.
2. Right-click the new environment and choose Rename from the context menu.
3. Type the name of the environment in the text field.
4. Create a Logical Host for the new environment. See Figure 11.

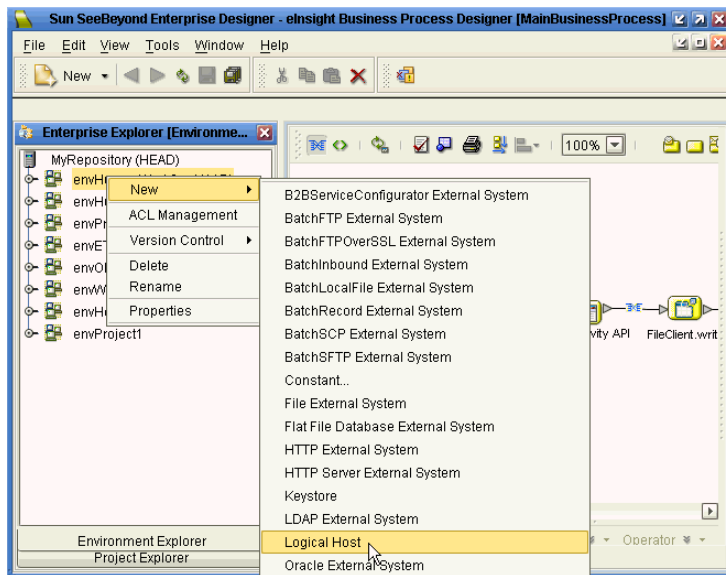


Figure 11 Creation of Logical Host

5. Create an Integration Server. See Figure 12.

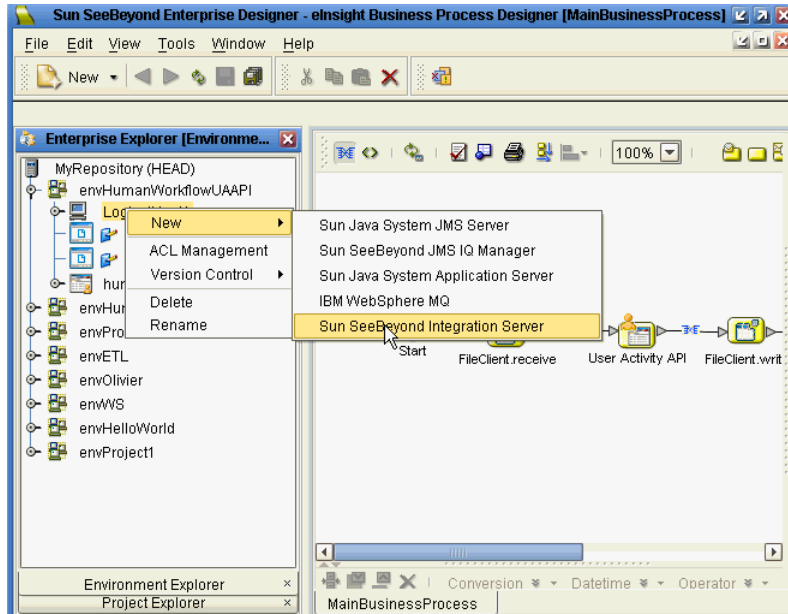


Figure 12 Creation of Integration Server

Configuring the Environment

Configure the environment:

1. Specify the password of the Integration Server by right-clicking the Integration Server and choosing Properties from the context menu.
2. Right-click the environment and choose WLM Connector External System to add a WLM viewer for connecting to the WLM in Java CAPS.
3. Specify a database for the WLM Connector: Click the Properties tab and edit the settings in the Properties dialog box, as appropriate. See Figure 13.

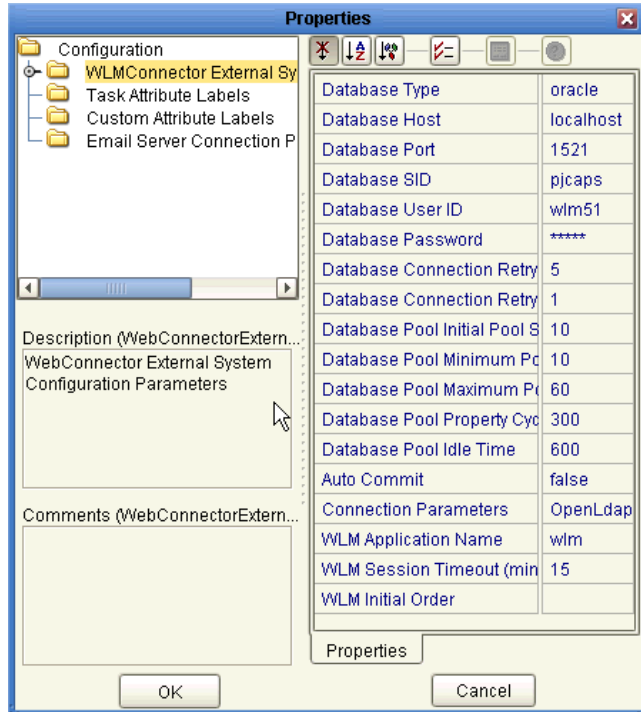


Figure 13 Properties for WML Connector Database

4. Right-click the environment and choose File External System from the context menu to define two file external systems.
Java CAPS prompts you to specify a name, for example, `extFile1`, for each of the systems.
5. Right-click the name of each of the systems and choose Properties from the context menu.
A Properties dialog box is displayed, in which you can specify the path of a file from which Java CAPS outputs the input. See Figure 14.

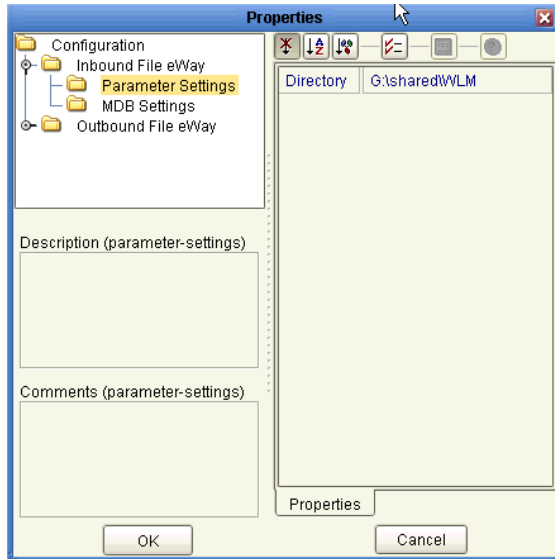


Figure 14 Location for Output File

6. Right-click the project and choose Connectivity Map from the context menu to connect the logical and physical locations of the resources that are in use.
For details, see the Java CAPS documentation.
7. Drag and drop the Main Business Process into the Connectivity Map and click the Connectivity Map Generator icon at the top. See Figure 15.

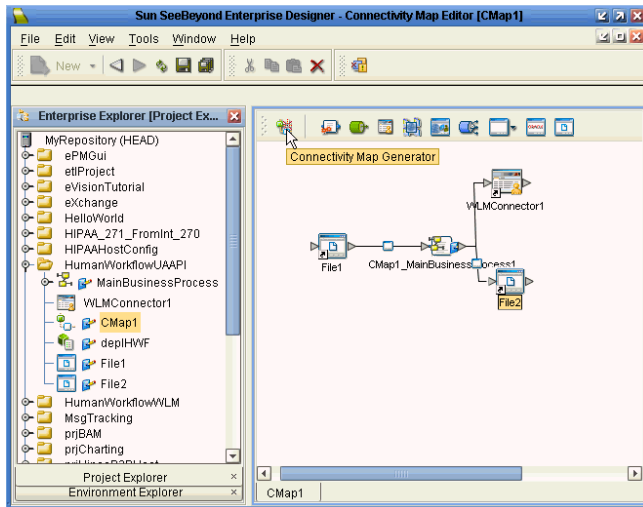


Figure 15 Generation of Connectivity Map

- Specify the input and output file names: Right-click the pink nodes in the links in the Connectivity Map to display the Properties dialog boxes.

Figure 16 shows the Properties dialog box for the input file.

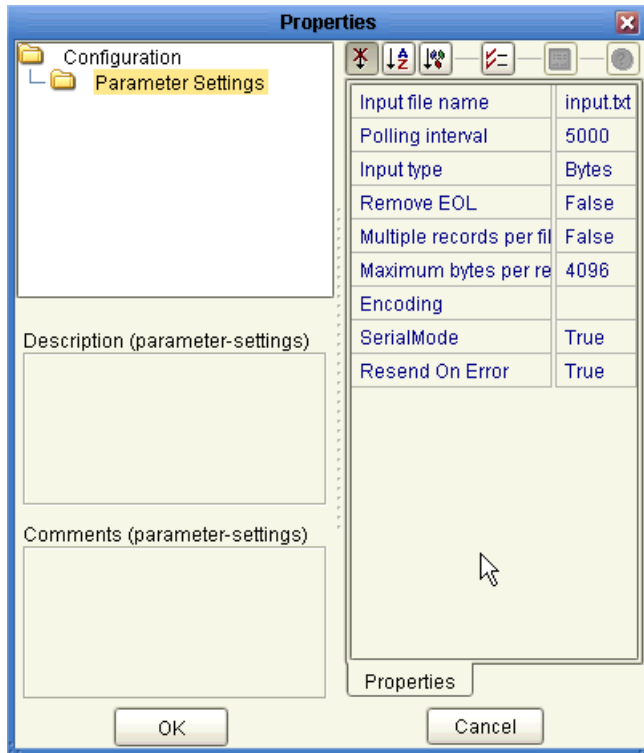


Figure 16 Properties for the Input File

Building and Deploying the Project

Finally, build and deploy the project:

- Right-click the project and choose Deployment Profile from the context menu.
- Select the Connectivity Map you just created as the default option for the deployment profile.
- Click Automap. See Figure 17.

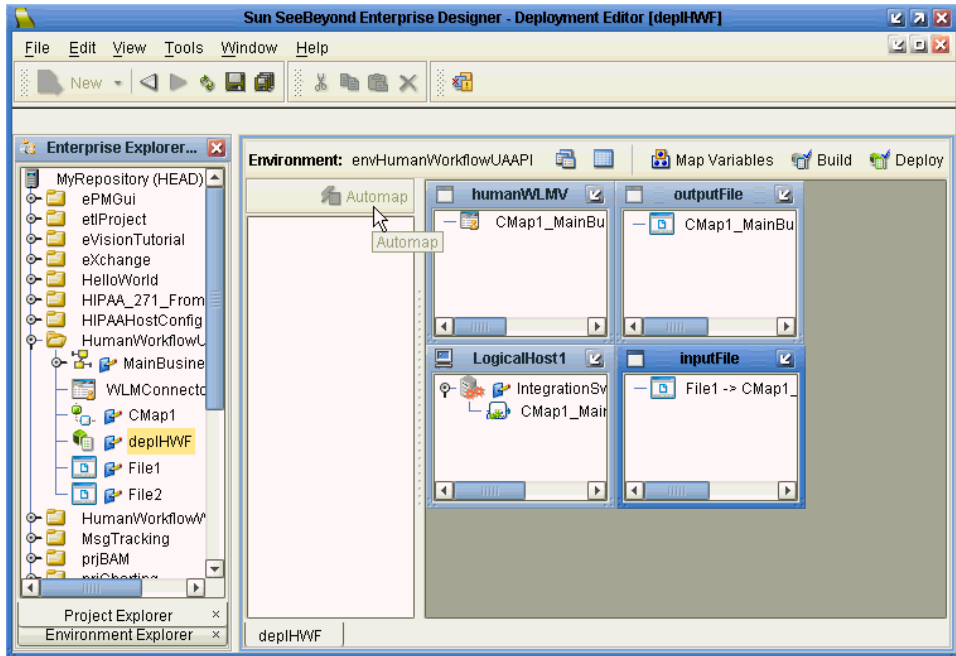


Figure 17 Creation of Deployment Profile

4. Download the `Workflowservice.jar` file from the Java CAPS repository and deploy the file in the Integration Server.

For details, see the section on the configuration of workflow service in the Java CAPS eInsight documentation.

5. Generate client stubs from the `Workflowservice.jar` file as `WorkflowServiceClient.jar`.

For details, see Appendix D of the *Sun Java CAPS eInsight User Guide*. That manual is part of the Java CAPS repository.

6. Place the stubs in the portlet that accesses the WLM.
7. Build and deploy the project.

Creating a Portlet for Managing Tasks

The Java CAPS task management protocol mandates that for a task to be considered complete, you must check out and execute that task.

You can create a simple portlet to do the following:

- Obtain a list of the tasks.
- Show the owners the tasks
- Execute the tasks.

Subsequently, Java CAPS signals the Business Process to execute the process.

Understanding the Execution Process

The execution process is as follows:

1. The portlet makes a Enterprise JavaBeans (EJB) remote call to `WorkflowService.jar`, which you deployed earlier.
2. `WorkflowService.jar` exposes the functions (APIs) that can manipulate the tasks. From those APIs, you can glean the details of the tasks requested by the end users.
For details on `WorkflowService.jar`, see the *Sun Java CAPS eInsight User Guide*.
3. Given the details, Java CAPS takes customized action to perform the tasks. The output of the actions matches the output to which the Business Process is linked.
An example: The User Activity API in the Business Process is linked to another Business activity and a configured rule mandates the transfer of X output of a User Activity node to Y input of the next Business activity node. The output generated from an execution of User Activity seamlessly handles the mapping. No worries for erroneous situations that might arise because of inadequate output. See Figure 18.

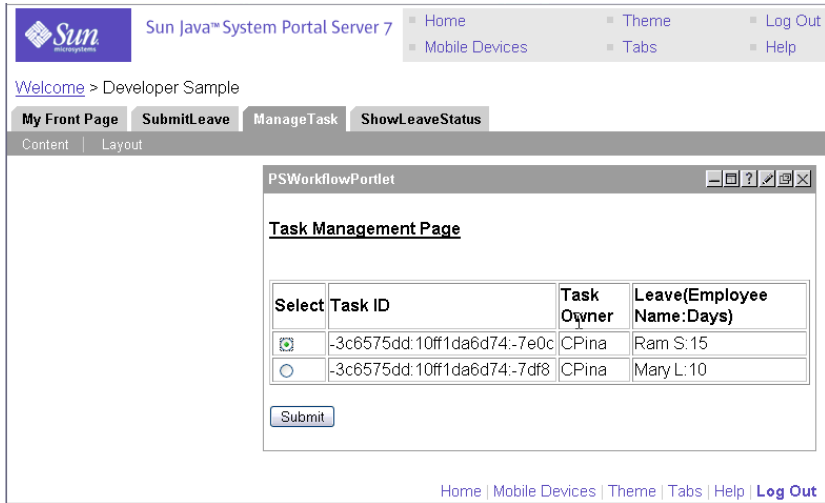


Figure 18 Task Management in Action by Portlet

Taking the Steps

To accomplish the preceding process, follow these steps:

1. Send the input with the Submit Input Portlet.
2. Manage the tasks with the Task Management Portlet.
3. Approve the task or process the data received as input.

The Business Process extracts the data from the input in step 1. The WLM Connector (defined in “Configuring the Environment” on page 12) and the Business Rule (defined in “Mapping Data” on page 7), ensure that the data goes into the task management database, where the same data awaits extraction by the Task Management Portlet through the `WorkflowService` API.

4. View the output in the Display Output Portlet.

Browsing the Sample Code

Here is a code segment of the sample portlet for managing tasks.

```
//Initialize Workflow Client with parameters such as hostname, port, etc.

Context ctx = new InitialContext(env);
Object obj = ctx.lookup(serviceJndi);
WorkflowServiceHome home = (WorkflowServiceHome) PortableRemoteObject.narrow(obj,
WorkflowServiceHome.class);
wfs = home.create();

//Use the following APIs to authenticate, set filters, perform operations on tasks and save them
back to the task information.
try {

    if(wfs.authenticateUser("Task Perofmer",passWord))
    {
        ArrayList taskIdList = new ArrayList();
        taskIdList.add(taskId);
        //Set the task filter. Here, you set it to all pending tasks.
        TaskFilter myFilter = new
TaskFilter(TaskFilter.TaskFilterType_ByTaskIdList,taskIdList);
        //Get tasks for a particular user with respect to the filter set above.
        ArrayList taskList = (ArrayList)wfs.getTasksForUser("UserName",myFilter);
        Iterator taskIterator = taskList.iterator();
        while(taskIterator.hasNext()) {

            //Get the task and set the owner of the task with APIs from Java CAPS on the task.
            Java CAPS Workflow requires steps to be in a particular order.
            //Checking out the task.
            if(wfs.checkoutTasks("User Name",myFilter)) {

                //Set attributes of the task with APIs from Java CAPS on the task and then
                save the tasks.
                wfs.saveTasks("User Name",myFilter);
                //Complete the task by committing to the Workflow Service.
                if(wfs.completeTasks("User Name",myFilter)) {
                    return true;
                }else {
                    return false;
                }
            }

        } //End of While Loop

    } //End of user authentication
    else{
        //Authentication failed for the end user
    }
} catch(Exception e) {
    e.printStackTrace();
}
```

Conclusion

Through an integration of Portal Server and Java CAPS and with minimal programming, human workflow tasks on portals can proceed accurately, smoothly, and intuitively. That's an innovative yet thoroughly tested approach well worth checking out.

References

- Sun Java System Portal Server
 - [Main page](#)
 - [Developer home](#)
 - [Software Forum](#)
- Weblogs
 - [The Portal Post](#)
 - [The Aquarium](#)
- Sun Java Composite Application Platform Suite
 - [Main page](#)
 - [Developer home](#)
- Related open-source software
 - [NetBeans Portlet Plug-in](#)
 - [Open LDAP](#)
 - [Portlet Container](#)
- Sun developer services
 - [Support](#)
 - [Training and certification](#)