

**Design Documentation for
JaWE2Openflow Project**

(Version 1.3)

August 17, 2004

N. Mehta

R. H. Barter

Developed by:

CIGNEX

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Neither CIGNEX Technologies Inc., nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

Software and documentation developed for LLNL by:



<http://www.cignex.com>

Contact information for LLNL and CIGNEX is provided below:

S/N	Company	Name Role/Dept	Phone	Email
1	LLNL	Bob Barter Project Manager	925-422-5150	barter1@llnl.gov
4	CIGNEX	Munwar Shariff CTO	408-718-1886	munwar@cignex.com
5	CIGNEX	Nisha Mehta Consultant and Project Leader	408-501-0455 x306	nisha@cignex.com

Document Revision History

Name	Date	Reason For Changes	Version
Nisha Mehta	6/11/2004	First Release	1.0 Draft
Nisha Mehta	7/2/2004	In this release, <ul style="list-style-type: none">▪ Changed document according to Bob Barter's comments▪ Added section for How to maintain the product	1.0
Stacy Peterson	7/29/2004	Formatted document for LLNL use. Edited text.	1.1
Bob Barter	7/29/2004	Minor changes	1.2
Bob Barter	8/17/04	Corrected contact information and added Cignex disclaimer	1.3

Table of Contents

1	INTRODUCTION	6
1.1	PURPOSE	6
1.2	SCOPE	6
1.3	OBJECTIVES OF THE DOCUMENT	6
1.4	ACRONYMS AND KEY TERMS	6
2	OVERVIEW.....	7
2.1	OVERVIEW	7
2.2	PURPOSE	7
3	ZOPE OPENFLOW	8
3.1	INTRODUCTION	8
3.1.1	<i>Activity-Based Workflow</i>	8
3.1.2	<i>Entity-Based Workflow</i>	9
3.2	REASONS TO USE OPENFLOW	9
3.3	THE PROCESS DEFINITION	10
3.3.1	<i>Activities (What)</i>	10
3.3.2	<i>Transitions (When)</i>	12
3.3.3	<i>Applications (How)</i>	12
3.3.4	<i>Users and Roles (Who)</i>	13
3.4	THE PROCESS INSTANCE.....	13
3.4.1	<i>Workitems</i>	13
3.4.2	<i>Worklist and Assigning Work</i>	14
3.4.3	<i>Exception Handling</i>	14
3.4.4	<i>Dynamic Redesign</i>	14
4	JAWE.....	15
4.1	OVERVIEW	15
4.2	PACKAGE	16
4.3	PROCESS	19
4.4	APPLICATIONS	21
4.5	DATA FIELDS (WORKFLOW RELEVANT DATA).....	24
4.6	PARTICIPANTS.....	27
4.7	ACTIVITIES	29
4.8	TRANSITIONS	33
4.9	COMPLETE JAWE PROCESS.....	35

5	JAWE2OPENFLOW PRODUCT	42
5.1	PREREQUISITES	42
5.2	HOW TO INSTALL	42
5.3	HOW TO GET HELP	42
5.4	HOW TO USE	42
5.4.1	<i>Login to ZMI</i>	42
5.4.2	<i>Create an Instance of the Product</i>	43
5.4.3	<i>Upload an XPDL file</i>	43
5.5	CHECK OUTPUT LOG FILE	43
5.6	HOW TO MAINTAIN THE PRODUCT	44
5.6.1	<i>JaWE2Openflow Product-Related Files</i>	44
6	JAWE TO OPENFLOW CONVERSION	45
6.1	PROCESS	45
6.2	APPLICATIONS	47
6.3	WORKFLOW RELEVANT DATA	49
6.4	PARTICIPANTS	51
6.5	ACTIVITIES	53
6.6	TRANSITIONS	60
7	SAMPLE CONVERSION PROCESS	64
7.1	SCENARIO	64
7.2	JAWE SCREEN FOR WORKFLOW.XPDL	64
7.3	WORKFLOW.XPDL (XML DUMP)	65
7.4	UPLOAD WORKFLOW.XPDL FILE	68
7.5	CUSTOMIZATION	80
8	SUMMARY	93
8.1	CONCLUSION	93
8.2	ASSUMPTIONS	93
8.3	LIMITATIONS	93

1 Introduction

1.1 Purpose

Lawrence Livermore National Laboratory (LLNL) has chosen CIGNEX Technologies, Inc. (CIGNEX) to design and develop the JaWE2Openflow conversion software. This document was created by CIGNEX as a project deliverable.

1.2 Scope

The scope of the project was to create a Zope Product, which reads the XPDL file (JaWE workflow process definition stored in the form of an XML file) and create an equivalent OpenFlow object in Zope.

1.3 Objectives of the Document

This document is intended to give the reader an overview of the JaWE2Openflow product and its components. Using the information in this document, a user will be able to create a workflow in JaWE, export it to XPDL and import it into OpenFlow.

1.4 Acronyms and Key Terms

Following are some of the acronyms used in the document.

Acronym/Term	Description
CIGNEX	CIGNEX Technologies, Inc.
CMF	Content Management Framework, Product by Zope
JaWE	Enhydra JaWE (Java Workflow Editor) is the first open source graphical Java workflow process editor fully according to WfMC specifications supporting XPDL as its native file format and LDAP connections. This is Object Web's Open Source middleware project. More information can be found at http://jawe.objectweb.org/
JaWE2Openflow	Name of the project
LLNL	Lawrence Livermore National Laboratory
OpenFlow	OpenFlow is an extremely flexible workflow engine, enabling to rapidly develop web based, and "workflow-oriented" applications on Zope.
OS	Operating System
Plone	Content Management System built on Top of Zope and CMF
ZODB	Zope Object Database
ZOPE	Zope is web server, web application server, database, search engine and content management framework, all in one available for free.

2 Overview

2.1 Overview

OpenFlow is an activity-based workflow management system. Activity-based means that the processes (workflows) are comprised of activities to be completed in order to get something done.

The main issue for a workflow management system is answering the question "who must do what, when, and how?" Activity-based workflow management systems like OpenFlow are designed to answer this question.

The process that defines the sequence of activities to be completed dictates what tasks should be done and when using the configured activity and transition definitions. An activity (the "what" part of the question) represents something to be done (i.e., giving authorization, updating a database, sending an e-mail, loading a truck, filling out a form, printing a document). Transitions define the appropriate sequence of activities for a process (the "when" part of the question).

Each activity will have an associated application designed to carry out the task (the "how" part of the question).

The answer to the "who" part of the question is determined by the user assigned to complete the activity. The assigned user is usually a person, but in some cases, activities are assigned to an automated system.

JaWE is a visual tool for creating, managing, and reviewing process definitions. In a straightforward and simple way, JaWE lets users quickly create workflow process definitions, check them, and store them for future use. Once a process definition is proven valid, it can be imported into new definitions, thus shortening the time and effort needed to define a workflow process.

JaWE2Openflow is a tool designed to convert JaWE workflows into Zope OpenFlow.

2.2 Purpose

The JaWE2Openflow product creates OpenFlow processes using an XPDL file. It defines applications, processes, activities, transitions between those activities, and participants for the workflow process. A page template named "index_html" is created for the datafields as defined in the XPDL file.

3 Zope OpenFlow

3.1 Introduction

Workflows can be divided into two basic categories: entity-based and activity-based. Activity-based workflow systems have workflow processes comprised of activities to be completed in order to accomplish a goal. In entity-based workflows, the focus is set on a document and the stages it must go through in order to be completed. OpenFlow is an activity-based workflow management system.

3.1.1 Activity-Based Workflow

The process describing research in a library can be viewed as an activity-based process.

An individual must complete the following steps:

1. Go to the library.
2. Locate relevant books using either the computer catalogs or the librarian's help.
3. Identify the most useful books on the topic.
4. Check the books out using the proper procedures for the library.

This list of activities must be carried out in order to obtain research books from a library. There is no main document in this process.

The fund request submission process might represent another example of an activity-based process. A funding request must go through the following steps:

1. The requestor will fill out a form specifying the intended purchase and how much it will cost.
2. The requestor will submit the completed form to the appropriate administrative organization.
3. The administrative organization will log and review the form.
4. The administrative organization will route the form to a financial/budgeting department to see if appropriate funds are available.
5. The financial/budgeting department will check the status of funds.
6. The financial/budgeting department will return the form and fund status information to the administrative organization.
7. The administrative organization will either approve or deny the request based on the information provided.
8. Upon approval, the administrative organization notifies the requestor, files the request and authorizes the release of funds.

9. The financial/budgeting department releases the requested funds and completes the necessary accounting tasks.
10. The requestor may purchase the approved item.

This process has a main document (the request form) but it has many side actions to be completed as well (i.e., signed authorization, routing to departments, filing, notifications). The best way to describe this is through an activity-based workflow, where the process is comprised of activities to be completed.

3.1.2 Entity-Based Workflow

Simpler processes often do not require the complex structure of an activity-based workflow management system and a simple entity-based workflow would suffice. For example, the publication of documents on a web site can be simply modeled by the document going through the states of new, submitted, and then approved or rejected. In entity-based workflows, the document is the main issue and its available actions are defined by its current status.

3.2 Reasons to Use OpenFlow

The process defining the sequence of activities to be carried out says **what** should be done and **when** by the definition of activities and transitions. An activity is the task (the **what**) that should be done (i.e., giving authorization, updating a database, sending an e-mail, loading a truck, filling out a form, printing a document). Transitions define the appropriate sequence of activities for a process and determine **when** activities will be done.

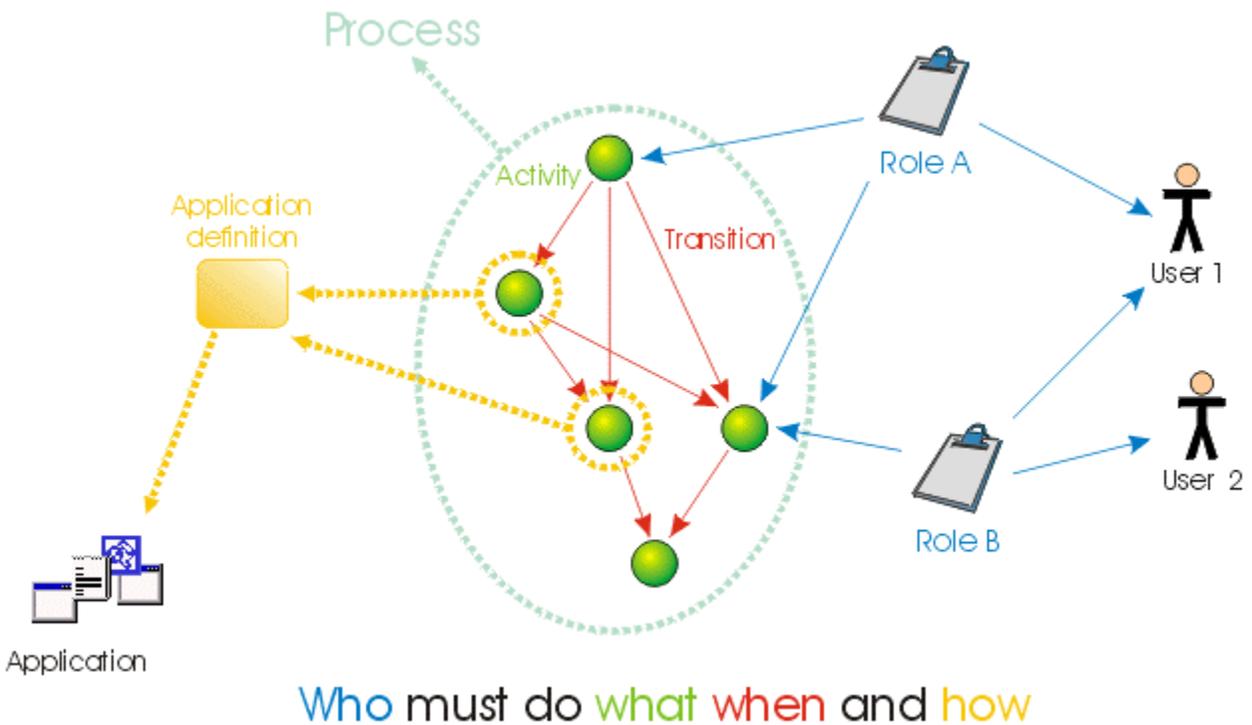


Diagram of workflow management system. Courtesy of OpenFlow (<http://www.openflow.it>).

Each activity has an associated application that defines **how** the activity will be carried out.

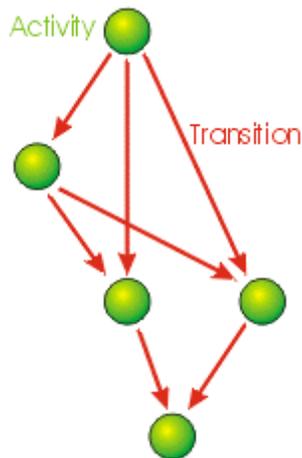
The user assigned to carry out the activity defines the **who** part of the equation. Usually a person will use activity applications, but sometimes the activity will be completed by an automated system.

Using a workflow management system has many advantages for an organization:

- Improves the efficiency and performance of processes. Many activities can be completed by an automated system. Updating databases, sending e-mail, compiling research, and archiving documents are examples of activities that a computer can complete without human intervention. This allows the activity to be completed faster while freeing a person's time for higher-level tasks.
- Increases organizational efficiency (always knowing the answer to "who must do what when and how?") Formalizing processes means not wasting time deciding what to do, having the right person doing each task, and ensuring each task will be completed appropriately.
- Enables managers to monitor the progress of the activities. For a given item, a project manager can know how far along the activities are in the process, what kind of transitions took place, and who completed each task.

3.3 The Process Definition

The goal of a process definition is to answer the question "who must do what, when and how?" Activities and transitions describe the **what** and **when** part of the process. The graphic below displays them using the "bubbles and arrows" model of what has to be done and when.



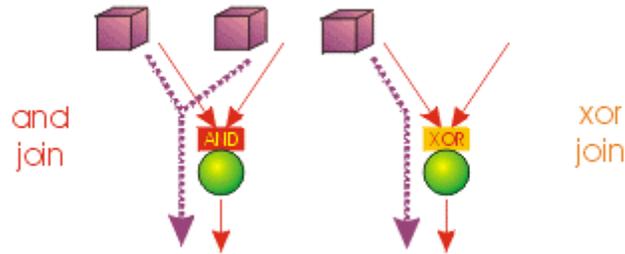
Applications describe the **how** part of the process and they are associated to activities. Users and roles describe the **who** part of the process.

3.3.1 Activities (What)

Each activity can be defined as one of three different types: application, sub-process, or routing. Usually each activity of the process definition describes something that has to be done, some kind of work. This work can either be carried out by an application or by a sub-process definition that more clearly defines the job to be done. Sometimes an activity's job will be handling the routing of work through the process. In this case, the activity is just a dummy activity that determines what should be done next.

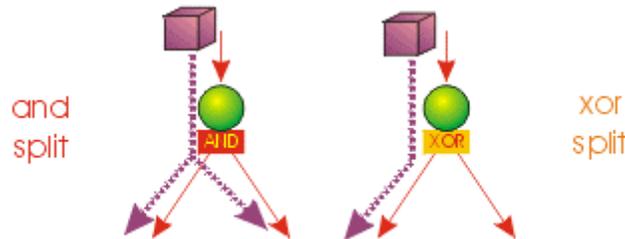
Each activity has one **incoming guard** for collecting incoming transitions. The activity incoming guard can be set to either “**and**” or “**xor**” for two different behaviors:

- A setting of “**and**” means that all the activities that lead to this activity must be completed in order to enable this activity to work.
- A setting of “**xor**” means that just one of the activities that lead to this activity must be completed in order to enable this activity to work.



Each activity also has one **outgoing guard** for collecting outgoing transitions. Like the incoming guard, the outgoing guard can be either set to “**and**” or “**xor**” for two different behaviors:

- A setting of “**and**” means that this activity will trigger the work of all connected activities, thus splitting the process flow into parallel (concurrent) flows.
- A setting of “**xor**” means that this activity will trigger the work of just one of the connected activities based on the evaluation of the condition (see Transitions).



Work performed by an activity will be triggered in different ways depending on the **start mode** setting of the activity itself. It can be set to one of two settings:

- **Automatic** start mode means that the activity will run its application as soon as an instance workitem arrives. There will be no worklist for users. OpenFlow itself will take care of starting the activity application on the workitem.
- **Manual** start mode means that OpenFlow will wait for user intervention to start the activity application. Usually, this is done through the call to the "callApplication" API of the workflow. This API should be called by the user’s worklist. The default worklist in OpenFlow does this.

Activity **finish mode** determines the way an activity transitions to the next activity (based on the transition condition). It can be set to one of two settings:

- **Automatic** finish mode means that as soon as a "completeWorkitem" API is called (usually by the activity application upon ending its work), the instance will be automatically forwarded to the next activity.

- **Manual** finish mode means that the user (or the activity application upon user input) must call the "forwardWorkitem" API. This enables the user to choose a transition. This mode is mainly reserved for manually steering the instance in alternate paths of the process. (An instance can be automatically steered by giving conditions to transitions.)

An activity also has **application** parameters (see Applications).

3.3.2 Transitions (When)

Transitions connect activities to each other. When connecting activity A with activity B, a transition states that as soon as A is finished B must be started.

Transitions can be guarded by conditions. A transition condition will be evaluated if the "from" activity has chosen one and only one path to be followed (i.e., the activity has an "xor" outgoing guard).

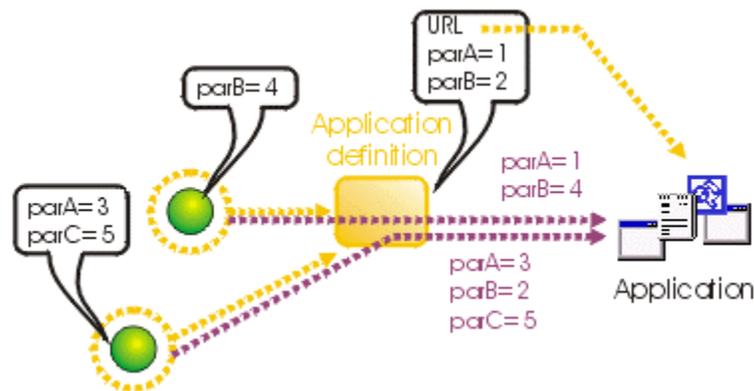
3.3.3 Applications (How)

An application is assigned to each activity in order to carry out the assigned activity. Applications can be anything triggered by a Uniform Resource Locator (URL) call. This includes Python scripts, DTML forms, SQL queries, Zope applications, or even external applications like Microsoft Word or other custom and dedicated applications.

The application is required to invoke the appropriate API for interacting with OpenFlow. For example the application will need to invoke the OpenFlow *completeWorkitem* API to signal OpenFlow that its job is finished and a new activity can be started.

A user can supply **parameters** to be passed to the application. These parameters can come from two sources:

- **Activity** parameters are passed to the application by its activity definition.
- **Application** parameters are specified in the application definition and will be used unless redefined by the activity parameters (therefore acting as defaults).



Some **standard parameters** passed to the application are:

- *openflow_id*—the identifier of the OpenFlow object.
- *process_id*—the identifier of the process definition.
- *activity_id*—the identifier of the activity that invoked the application.
- *instance_id*—the identifier of the process instance that triggered the activity.
- *workitem_id*—the identifier of the specific workitem that tracks the triggered activity work.

3.3.4 Users and Roles (Who)

Users are assigned to applications through roles. One user can be listed in one or more roles. Roles can list one or more activities. Each user will be able to work on all of the activities that are listed in the assigned roles.

Each role maintains three different lists:

- **Users** identifies users that are assigned to the role.
- **Enabled activities** identifies the process activities that users in this role can work on.
- **Assignable activities** identifies the process activities that users listed in this role can assign to other users that are enabled to work on them.

3.4 The Process Instance

A process definition gives the instructions for completing work, while a process instance is the execution of a process definition. For example, if the process definition describes how to submit a fund request, a process instance of that process definition is an actual submission for funding.

3.4.1 Workitems

Workitem is the execution of a single activity of the process definition. Workitems are created every time an activity is triggered and are never destroyed. When an activity completes its job, the workitem created to represent the execution of that activity is assigned a “complete” status. A process instance is a collection of workitems: one per activity executed (or in execution).

One of the most important tasks of a workitem is to keep track of events during the execution of its activity. Since the process instance keeps track of all its workitems, and each workitem keeps track of all the events that happen during the execution of the activity, the complete history of everything that has happened is recorded in each process instance.

This is often useful because an event log of every single instance begun in the process is available and an analysis of these logs can help improve process design (i.e., manage load balancing, monitor completion times).

3.4.2 Worklist and Assigning Work

Each user is associated with a **worklist**, a list of pending workitems for activities. The worklist is not set to a given size, but grows when activities require some work to be done and shrinks when the work on activities is completed.

Users are presented their worklists to let them know what is to be done. In OpenFlow, three policies exist for work assignments:

- The **pull** policy means that users will choose what to do, as if the work to be done was gathered in a common pool where users choose tasks.
- The **manual push** policy means the user is assigned work by another user.
- The **automatic push** policy means the user is automatically assigned work by the workflow engine.

Whether workitems are self-assigned or assigned by someone else, once a user is assigned a task, the user will be the only one enabled to carry it out. Other users will not be able to see the assigned task in their worklist.

3.4.3 Exception Handling

In existing workflow management systems, flexibility is a major issue because business processes are often revised and streamlined. The system must be able to handle changes to process definitions, even while they are running. It must also be able to handle exceptions when a user must handle an issue that was not foreseen in the process definition.

Instead of a normal handling of a workitem, the user can make it "fall out" of the normal process flow. The exception task will then be available to any user who is designated to handle exceptions. As soon as the workitem is fixed, it can be put back into any activity of the process to resume the process flow. In this way, an appropriately authorized user can change the workitem data to adjust to the situation.

3.4.4 Dynamic Redesign

A process definition can be changed while in execution (i.e., new activities can be added, old ones deleted, transitions created or modified). As soon as the process is changed all current process instances will read the new process definition and begin using it. Any invalidating situation will cause the appropriate workitem to be tagged as an exception and be routed to designated users.

Dynamic redesign and exception handling are closely related. Exception handling allows for dealing with unforeseen events and suggests changes in the process definition to handle that situation in the future. On the other hand, dynamic redesign may cause many process instances to register an invalid status and exception handling should be used to recover from these situations.

4 JaWE

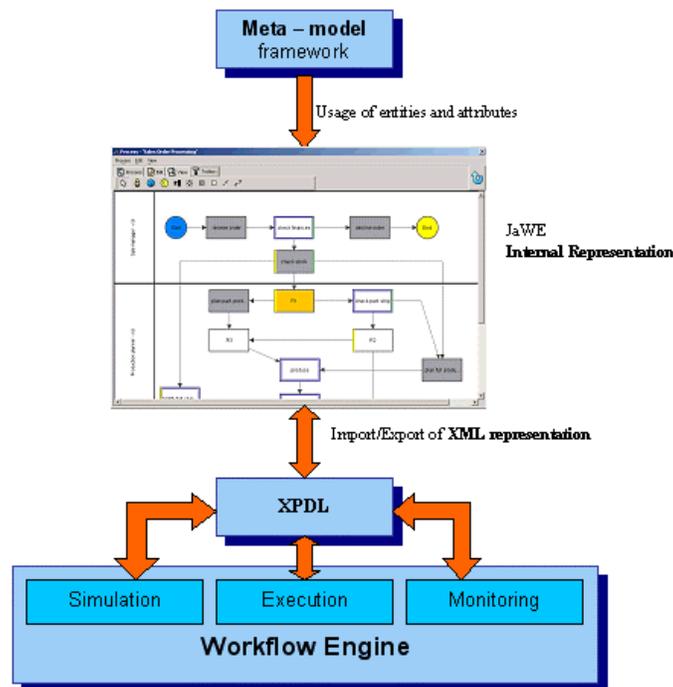
4.1 Overview

The description of how JaWE works corresponds to Workflow Management Coalition (WfMC) specifications (www.wfmc.org). WfMC provides an interface for defining workflow processes that uses a common meta-model for describing the process definition and XML schema for the interchange of process definitions called XML Process Definition Language (XPDL). This tutorial is focused on the XPDL Interface and how JaWE implemented it. Some parts of this document are taken from the original XPDL specification (WFMC-TC-1025).

JaWE is a tool for process definition modelling. The final output of this process modelling is a XPDL file, which can be interpreted at runtime by workflow engine. JaWE accomplishes three main goals:

- Graphical representation of process definitions.
- Export of process definitions to XPDL.
- Import of any valid XPDL and its graphical representation.

The workflow process definition interface defines a common interchange format, which supports the transfer of workflow process definitions between different products. A workflow process definition generated by JaWE can be interpreted by different workflow run-time products. The principles of process definition interchange are based on meta-model framework. This meta-data model identifies commonly used entities within a process definition, their relationships and attributes. A variety of attributes describe the characteristics of this limited set of entities. Using this meta-model, JaWE can transfer models using XPDL as a common exchange format. JaWE is also used for internal representation of process definitions.



A mandatory minimum set of objects must be supported within XPD. This "minimum meta-data model" identifies those commonly used entities within a process definition and describes their usage semantics. Extensibility is provided by the facility to encompass additional object attributes ("extended attributes") which can be included as extensions to the basic meta-model to meet the specific needs of an individual product or workflow system.

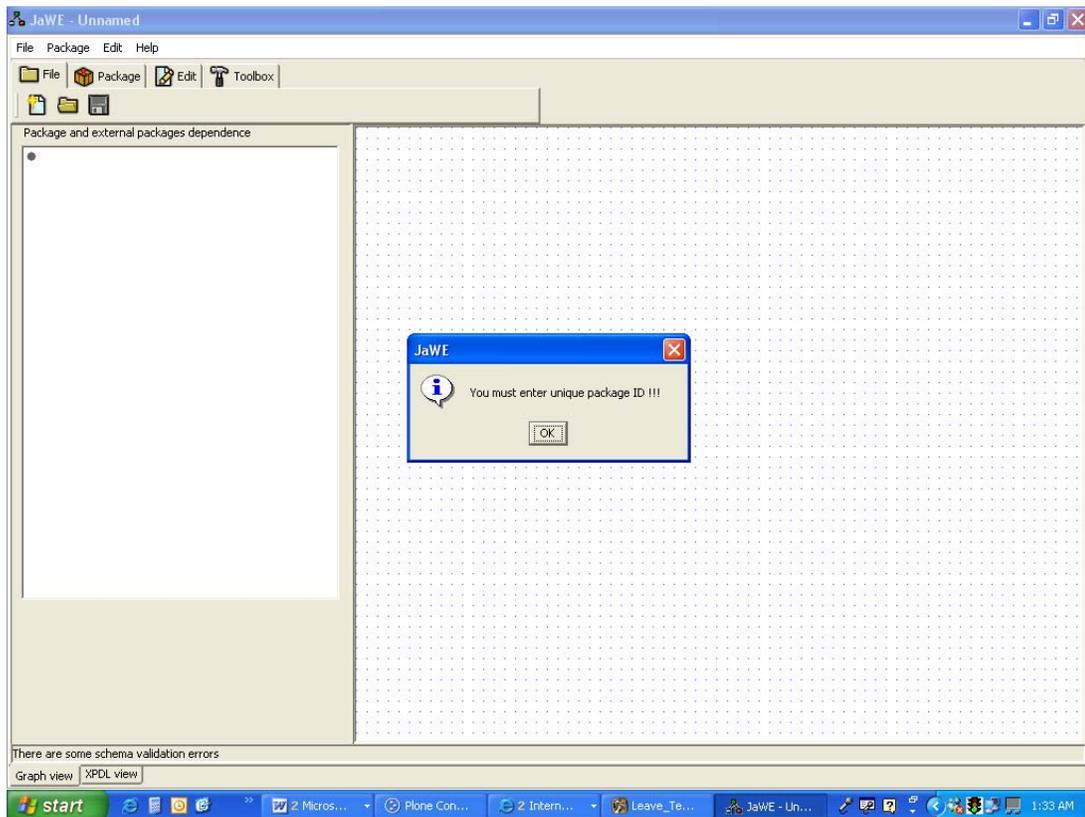
4.2 Package

Several processes that may share the same tools (applications) and participants can be defined within one package. The package acts as a container for grouping together a number of individual process definitions and associated entity data, which is applicable to all the contained process definitions.

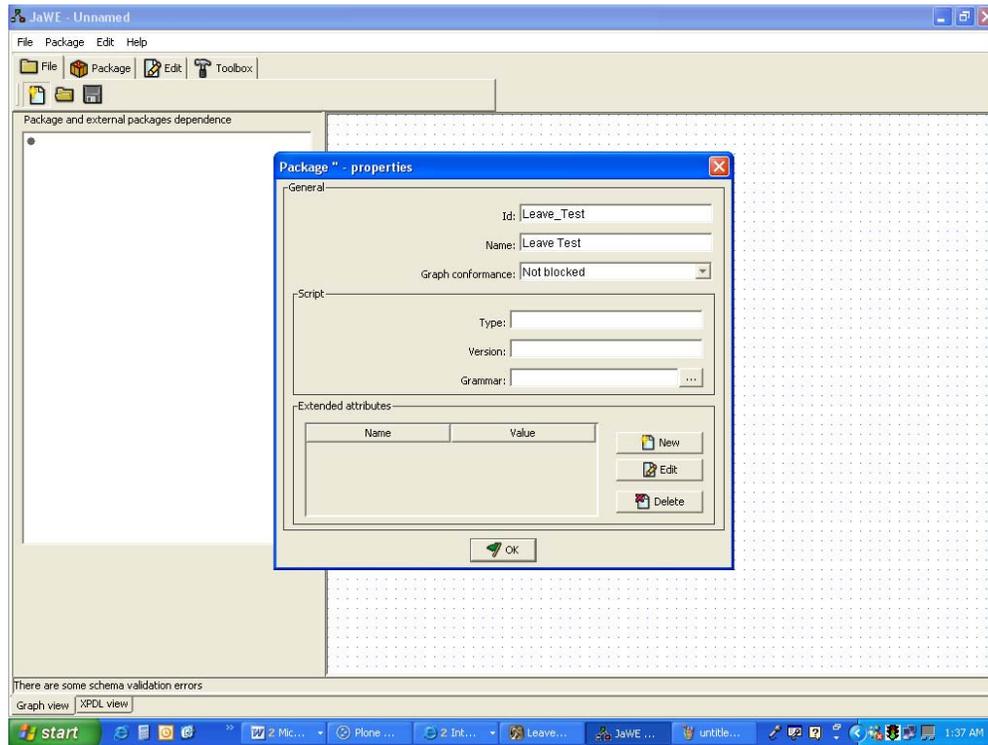
A JaWE XPD to Zope OpenFlow sample conversion example is explained in Section 7.

To Create a Package:

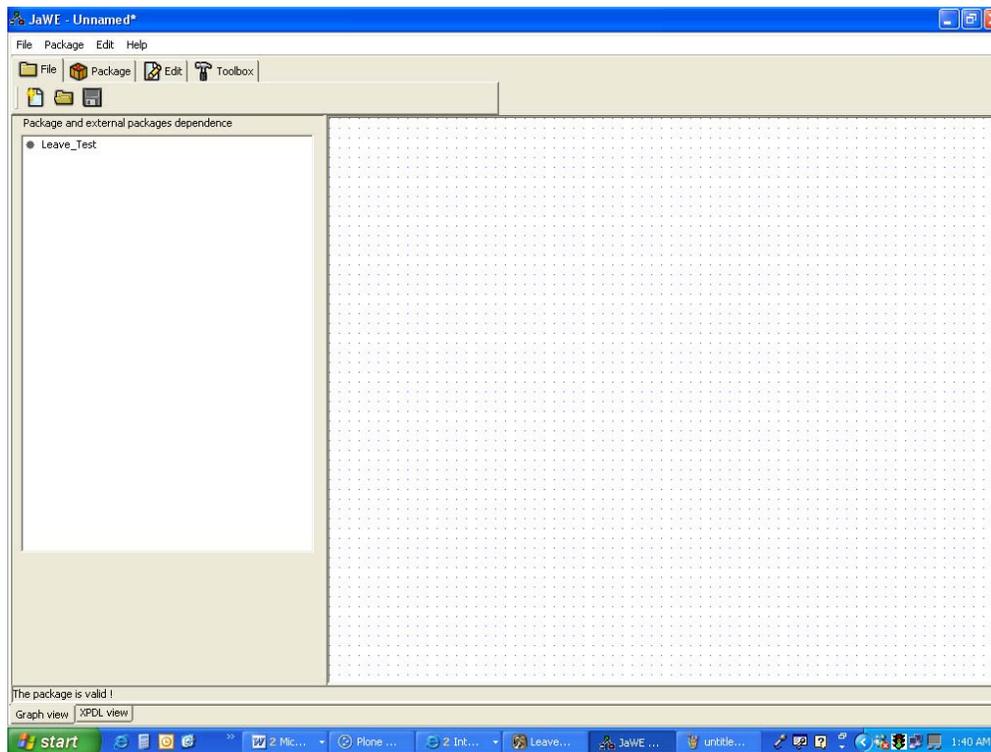
1. Open JaWE editor and select File – New. An information box displays stating the need for a unique package ID. Click on the OK button.



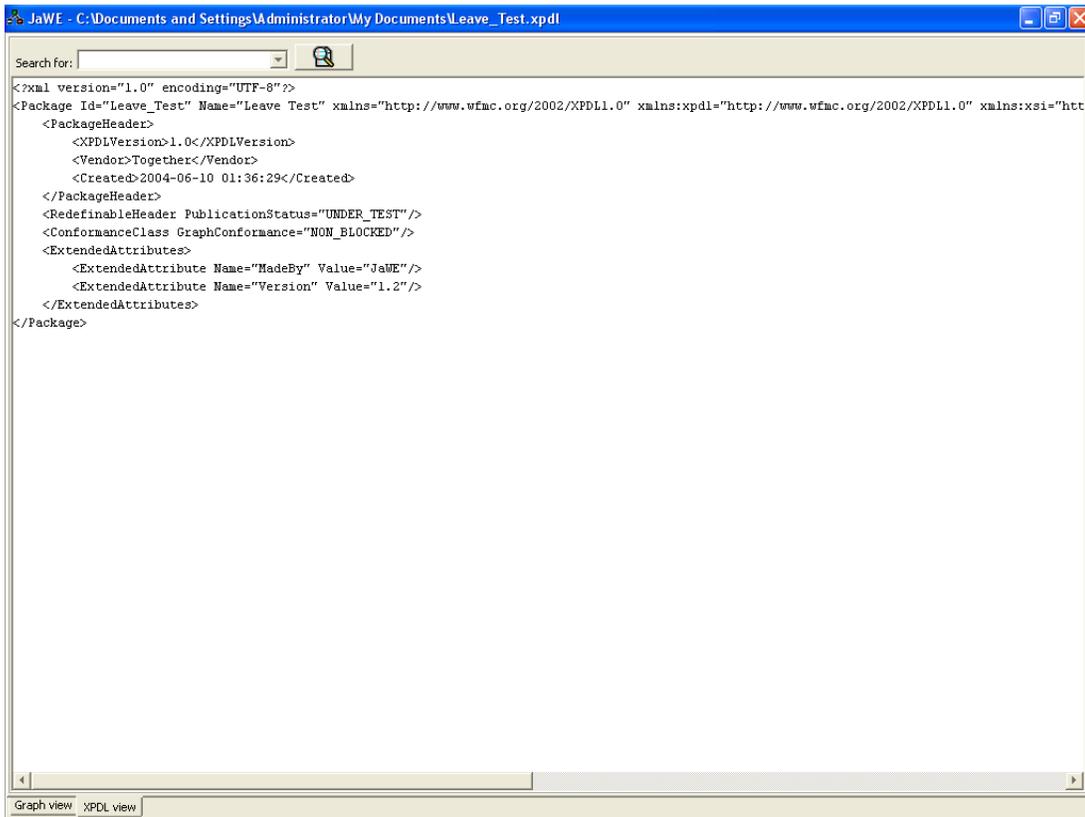
2. A package properties window is displayed. Enter a value for the ID field (required field) and name and click the OK button.



3. A package with the given 'ID' is created.



4. The following XPDL code is produced.



The screenshot shows a window titled "JaWE - C:\Documents and Settings\Administrator\My Documents\Leave_Test.xpdl". The window contains a search bar and a text area with the following XML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package Id="Leave_Test" Name="Leave Test" xmlns="http://www.wfmc.org/2002/XPDL1.0" xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0" xmlns:xsi="http://www.wfmc.org/2002/XPDL1.0" xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0 http://www.wfmc.org/2002/XPDL1.0/XPDL1.0.xsd">
  <PackageHeader>
    <XPDLVersion>1.0</XPDLVersion>
    <Vendor>Together</Vendor>
    <Created>2004-06-10 01:36:29</Created>
  </PackageHeader>
  <RedefinableHeader PublicationStatus="UNDER_TEST"/>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <ExtendedAttributes>
    <ExtendedAttribute Name="MadeBy" Value="JaWE"/>
    <ExtendedAttribute Name="Version" Value="1.2"/>
  </ExtendedAttributes>
</Package>
```

At the bottom of the window, there are two tabs: "Graph view" and "XPDL view", with "XPDL view" being the active tab.

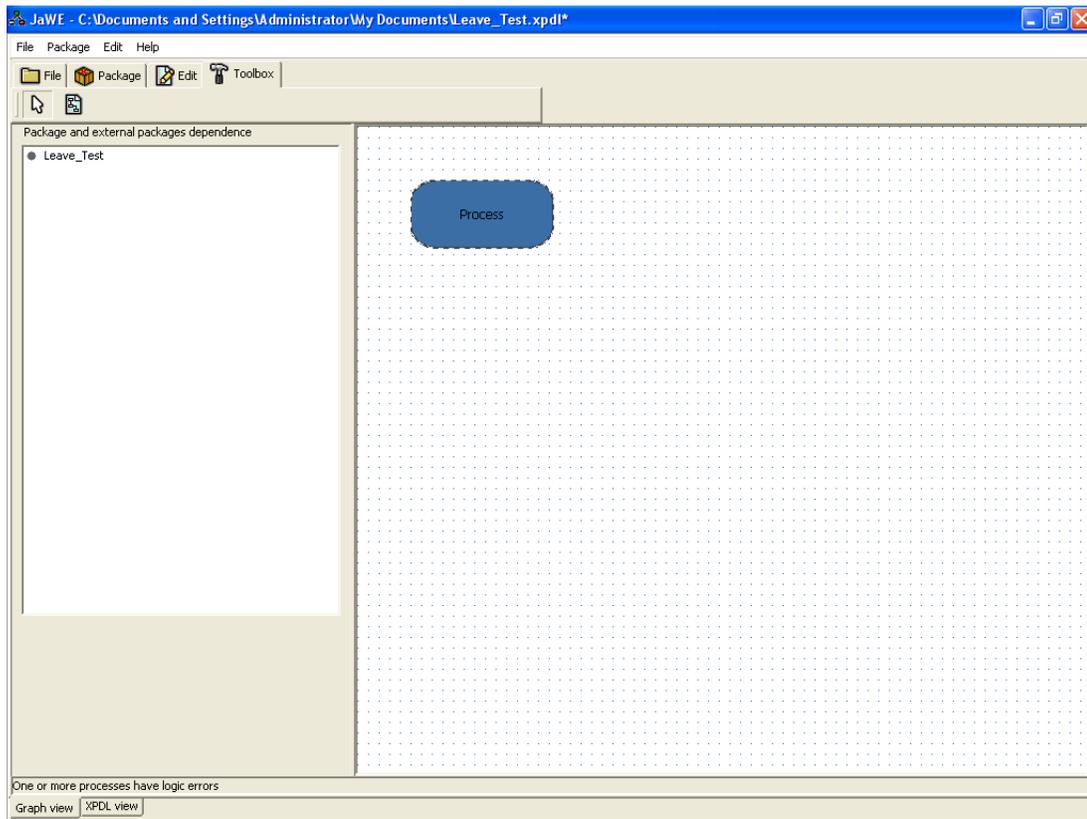
4.3 Process

The workflow process defines the elements that make a workflow. It contains definitions or declarations, respectively, for activity and, optionally, for transition, application, and process-relevant data entities.

A workflow process may run as an implementation of an activity of type subflow; in this case Process Toolbar provides all settings for Processes attributes in JaWE

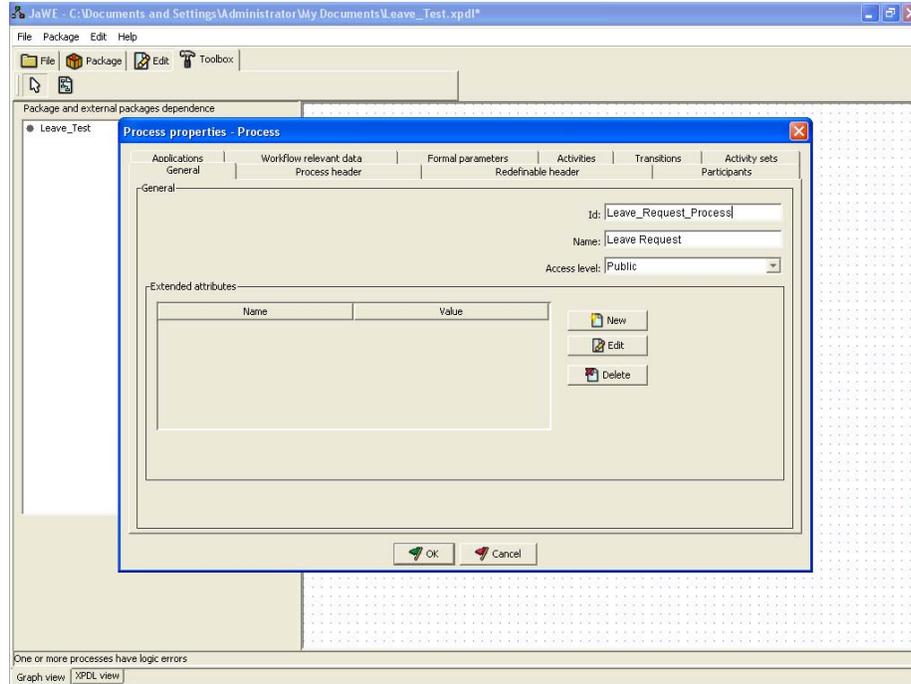
To Create a Process:

1. Click on the Toolbox tab and drag and drop the Insert Process icon onto the graph container. A process is created.

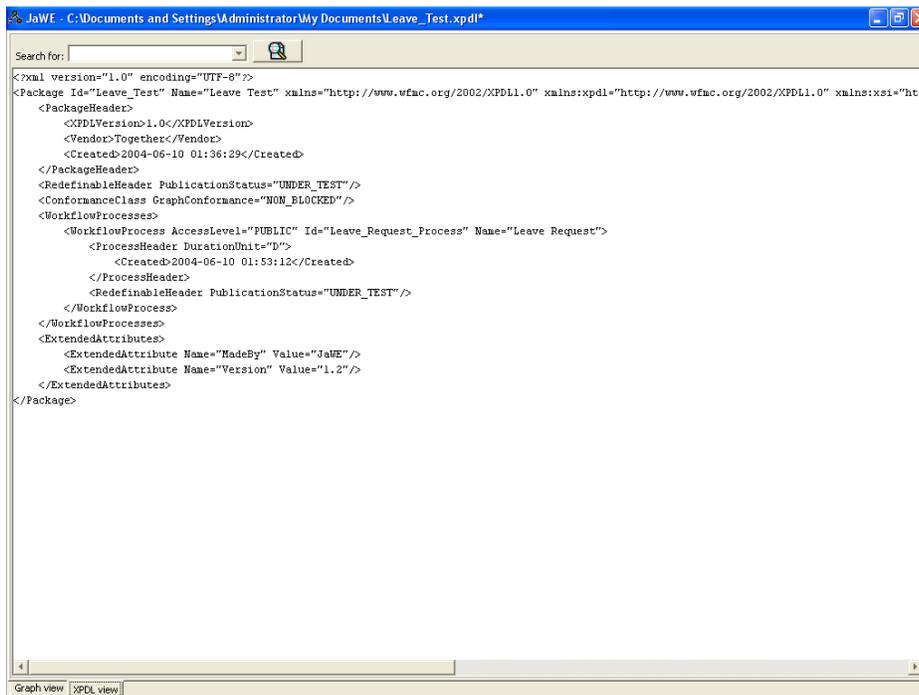


2. Double click on the process (or right click and select properties) to complete the ID and Name fields. Click the OK button. A process with the given name is created.

Return to this screen (right click the process and select Edit) to add participants, activities and transitions to the process (using the appropriate tabs).



3. The following XPD L code will be produced.

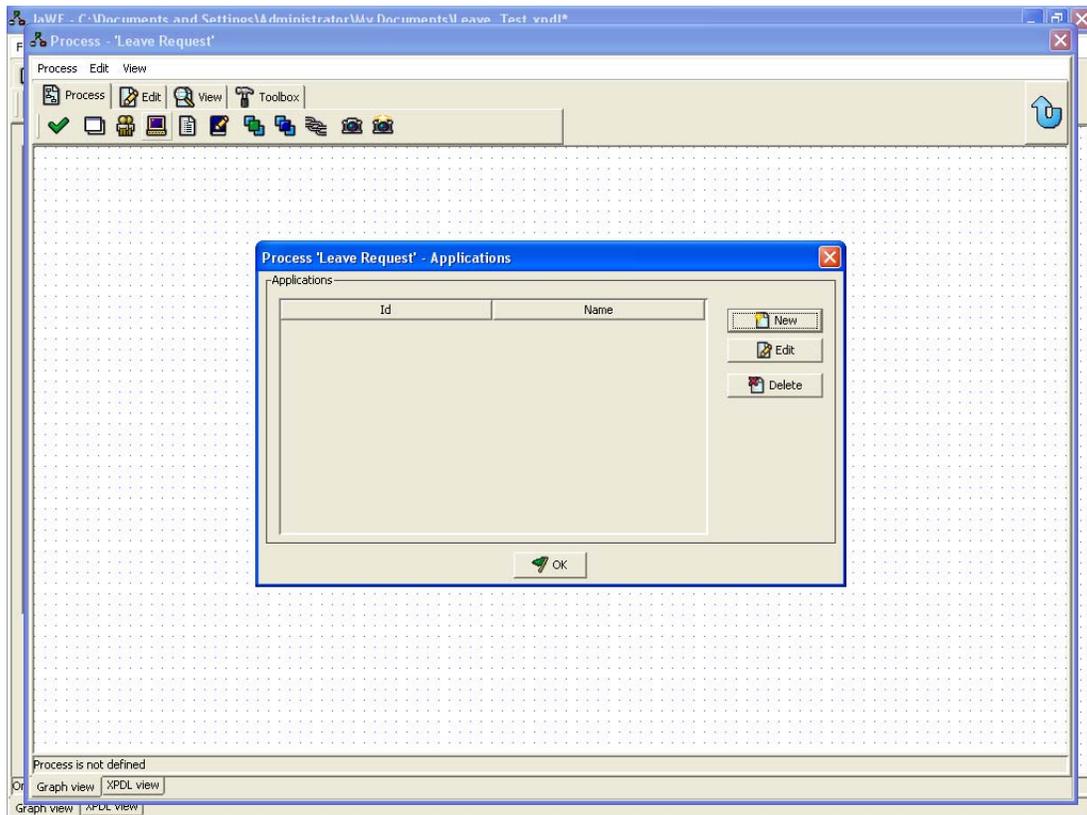


4.4 Applications

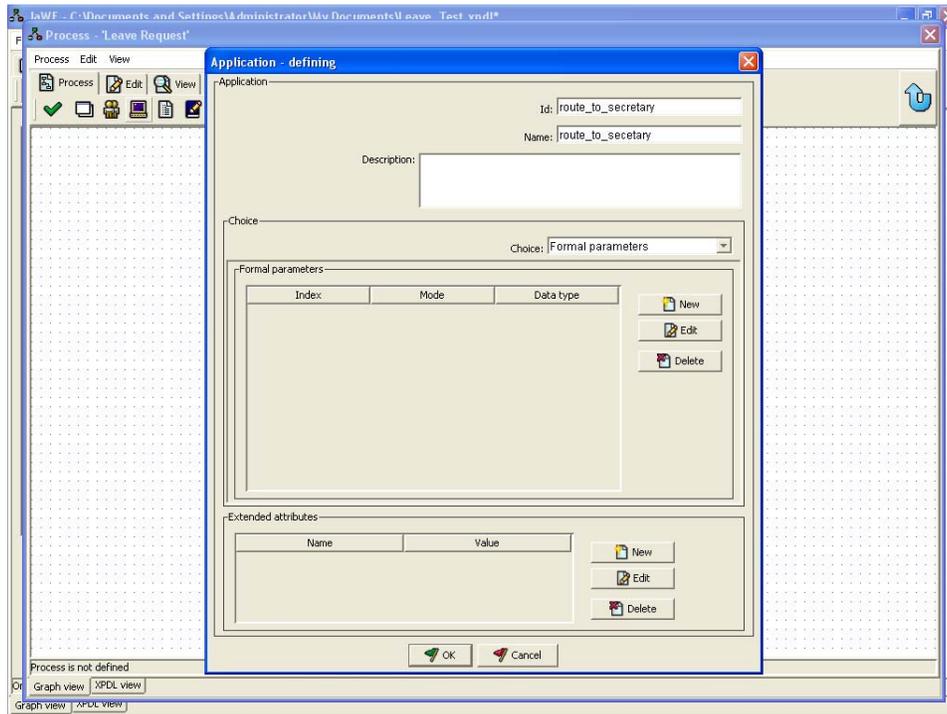
A workflow application declaration is a list of all applications or tools required and invoked by the workflow processes. Some tools are generic named tools (i.e., send_mail or scan_document). The real definition of the tools is not necessary because of the way they must be handled in multi-platform environments where a different program (or function) must be invoked for each platform.

To Create an Application:

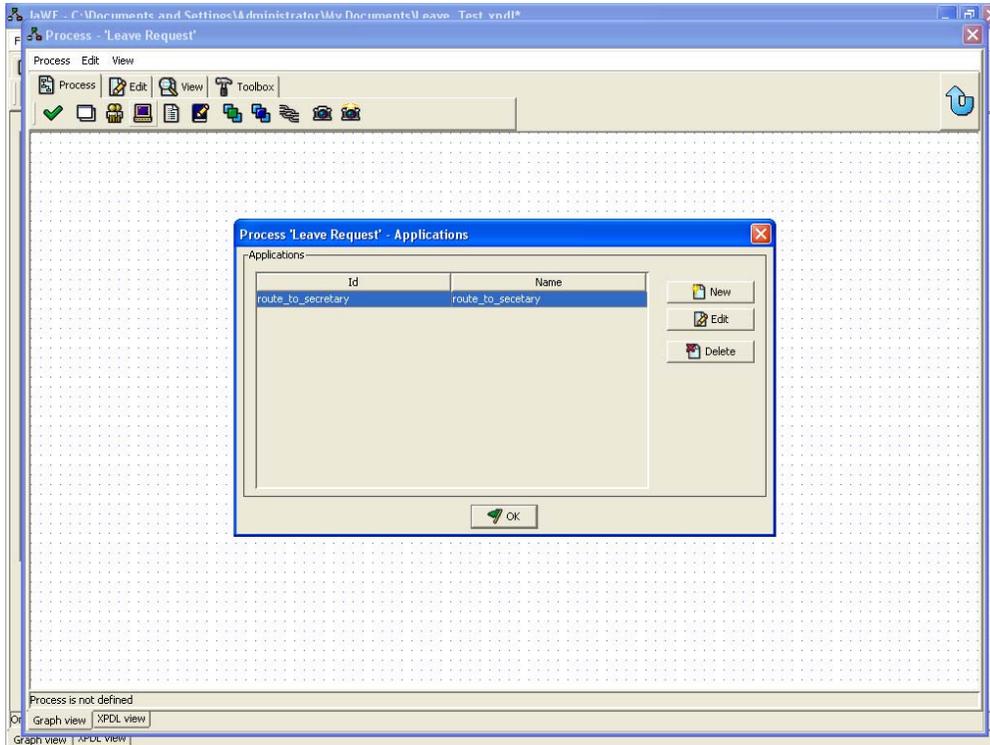
1. Click on the Applications icon at the process level. A Process – Applications window displays. Click on the New button and add an application.



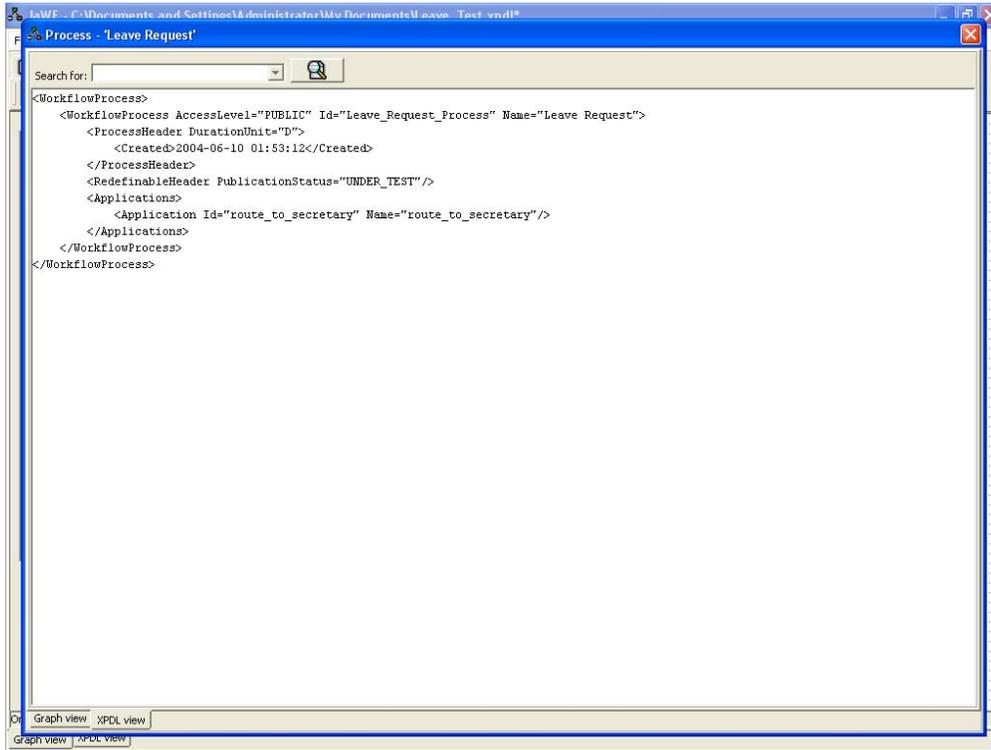
2. Complete the ID, Name and Description fields as appropriate for the application. Click the OK button.



3. An application is added.



4. The following XPDL code will be produced.



The screenshot shows a window titled "Process - Leave Request" with a search bar at the top. The main area displays the following XML code:

```
<WorkflowProcess>  
  <ProcessHeader AccessLevel="PUBLIC" Id="Leave_Request_Process" Name="Leave Request">  
    <Created>2004-06-10 01:53:12</Created>  
  </ProcessHeader>  
  <RedefinableHeader PublicationStatus="UNDER_TEST"/>  
  <Applications>  
    <Application Id="route_to_secretary" Name="route_to_secretary"/>  
  </Applications>  
</WorkflowProcess>  
</WorkflowProcess>
```

At the bottom of the window, there are tabs for "Graph view" and "XPDL view", with "XPDL view" currently selected.

4.5 Data Fields (Workflow Relevant Data)

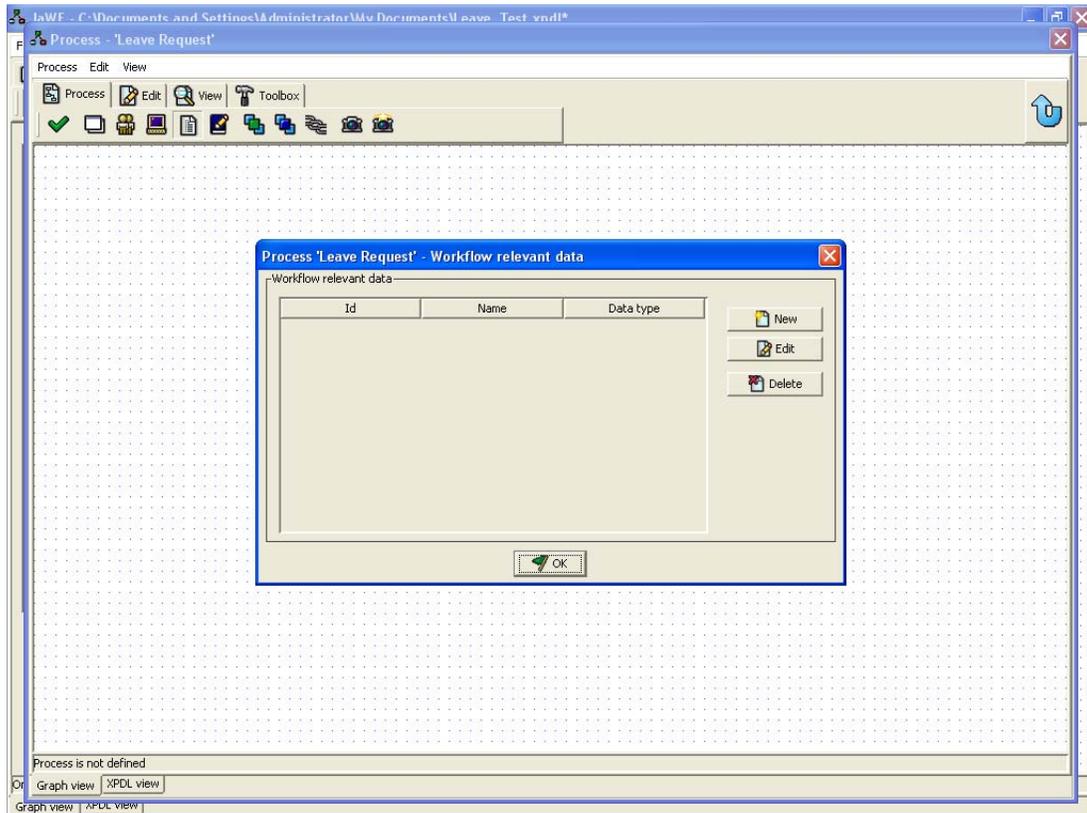
Workflow relevant data (known in XPDL as data fields) represent the variables of a process definition or package definition. They are typically used to maintain decision data (used in conditions) or reference data values (parameters) that are passed between activities or sub-processes. The workflow relevant data list defines all data objects that can be used within the workflow process. The attribute “type” explicitly specifies all information needed for a workflow management system to define an appropriate data object for storing data that is to be handled by an active instance of the workflow process.

Workflow relevant data can be defined in a workflow process (the workflow process relevant data) and in a package (the package relevant data). Workflow relevant data is used to create a workflow application home page (“index_html”). All fields defined using workflow relevant data in JaWE will be used to create an HTML form in the index_html page. After converting JaWE XPDL to Zope OpenFlow using this product, the user can click on the Submit button on the index_html page to create a workflow instance. The scopes differ in that the former may only be accessed by entities defined inside that process, while the latter may be accessed by entities inside any process defined within that model.

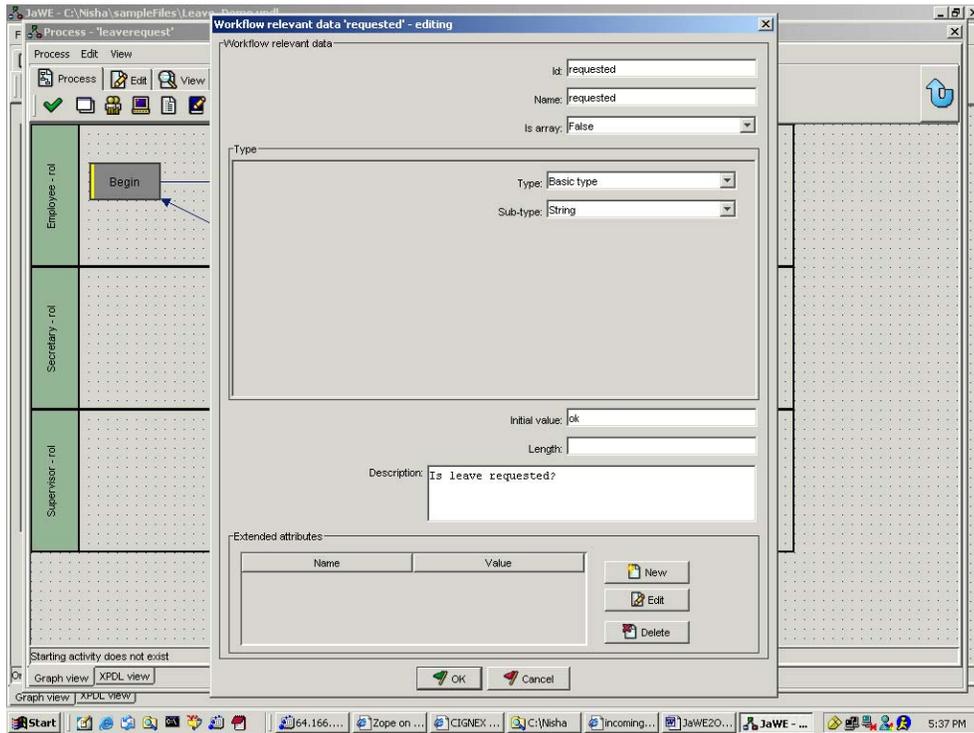
Workflow relevant data has a scope that is defined by the directly surrounding meta-model entity and is not nested. The visibility of its identifier is also defined by that entity.

To Create Data Fields:

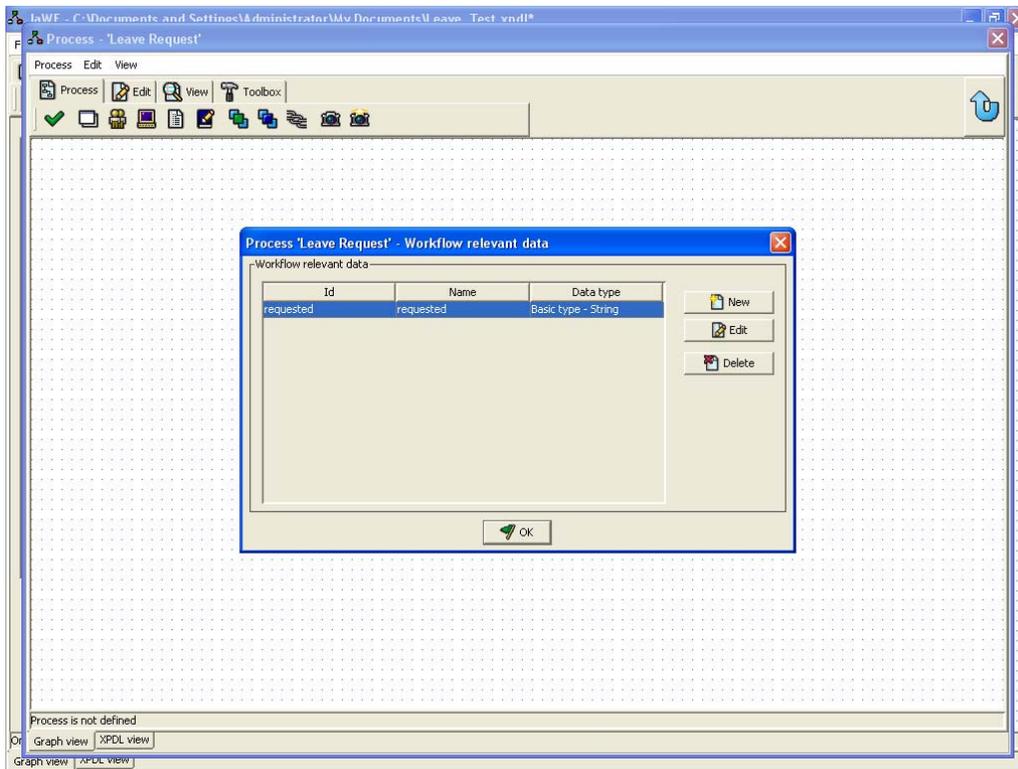
1. Click on the Workflow relevant data icon. A Workflow relevant data – Defining window appears. Click on the New button.



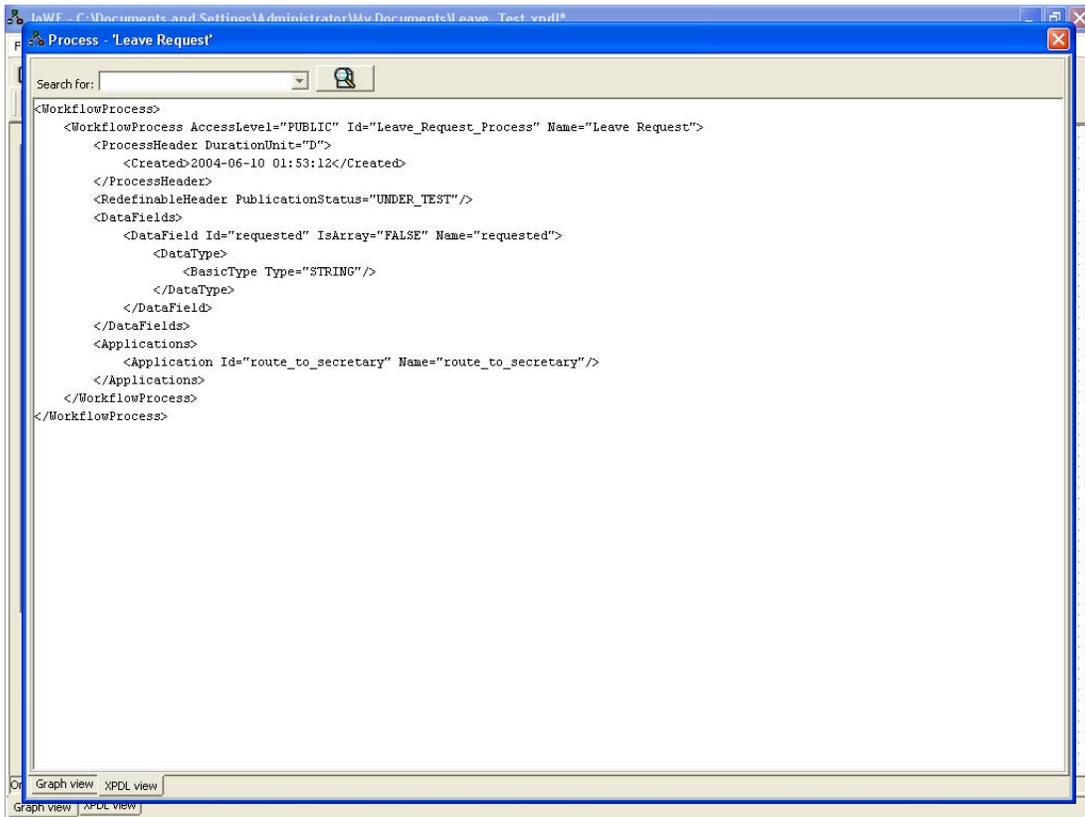
2. Enter the appropriate information in the ID, Name, Is Array, Type, and Sub-Type fields and click the OK button.



3. The workflow relevant data is created.



4. The following XPDL code will be produced.



```
<WorkflowProcess>  
  <WorkflowProcess AccessLevel="PUBLIC" Id="Leave_Request_Process" Name="Leave Request">  
    <ProcessHeader DurationUnit="D">  
      <Created>2004-06-10 01:53:12</Created>  
    </ProcessHeader>  
    <RedefinableHeader PublicationStatus="UNDER_TEST"/>  
    <DataFields>  
      <DataField Id="requested" IsArray="FALSE" Name="requested">  
        <DataType>  
          <BasicType Type="STRING"/>  
        </DataType>  
      </DataField>  
    </DataFields>  
    <Applications>  
      <Application Id="route_to_secretary" Name="route_to_secretary"/>  
    </Applications>  
  </WorkflowProcess>  
</WorkflowProcess>
```

4.6 Participants

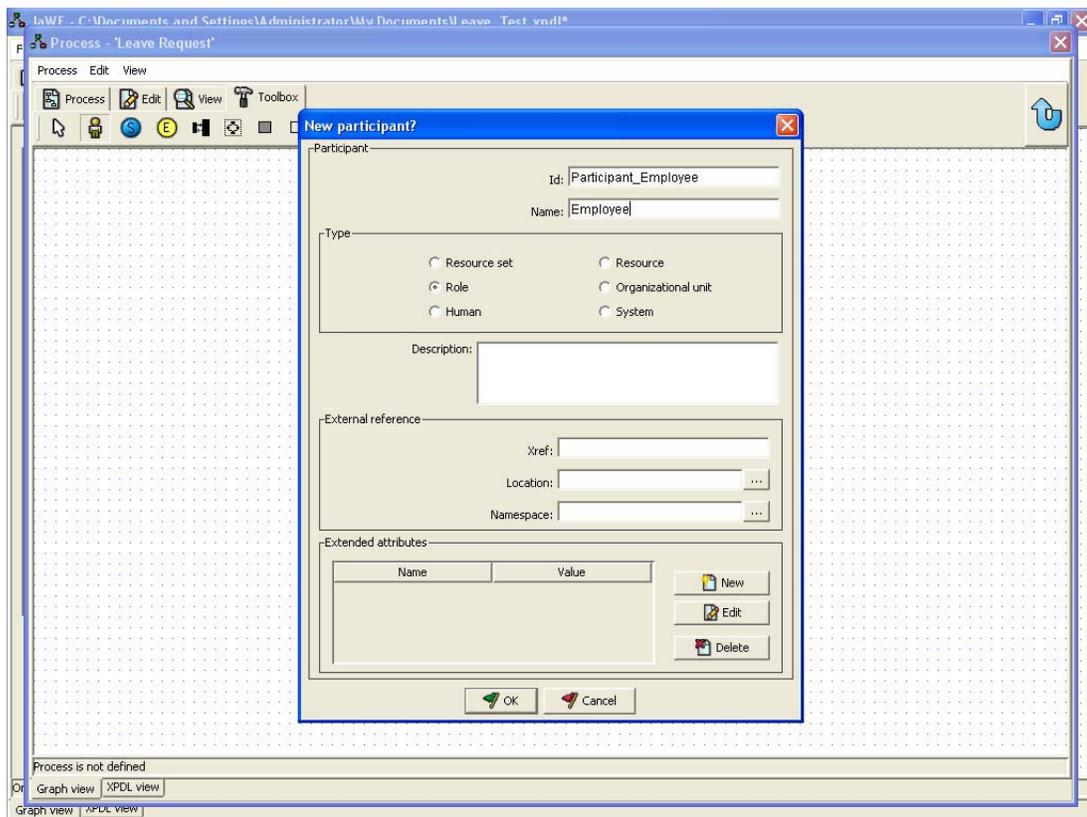
The WfMC meta-model specification defines a simple in-built (minimal) organizational model or permits access to an externally-defined organizational model. Participants in JaWE are part of a minimal organizational model. The connection with the organizational model is used to define activities (identifying an individual to complete an activity) and processes (identify an individual responsible for a process).

Workflow participants have a scope and visibility equivalent to their extended attributes. All referenced workflow participants must be defined in the scope where they are used, at least in the same package.

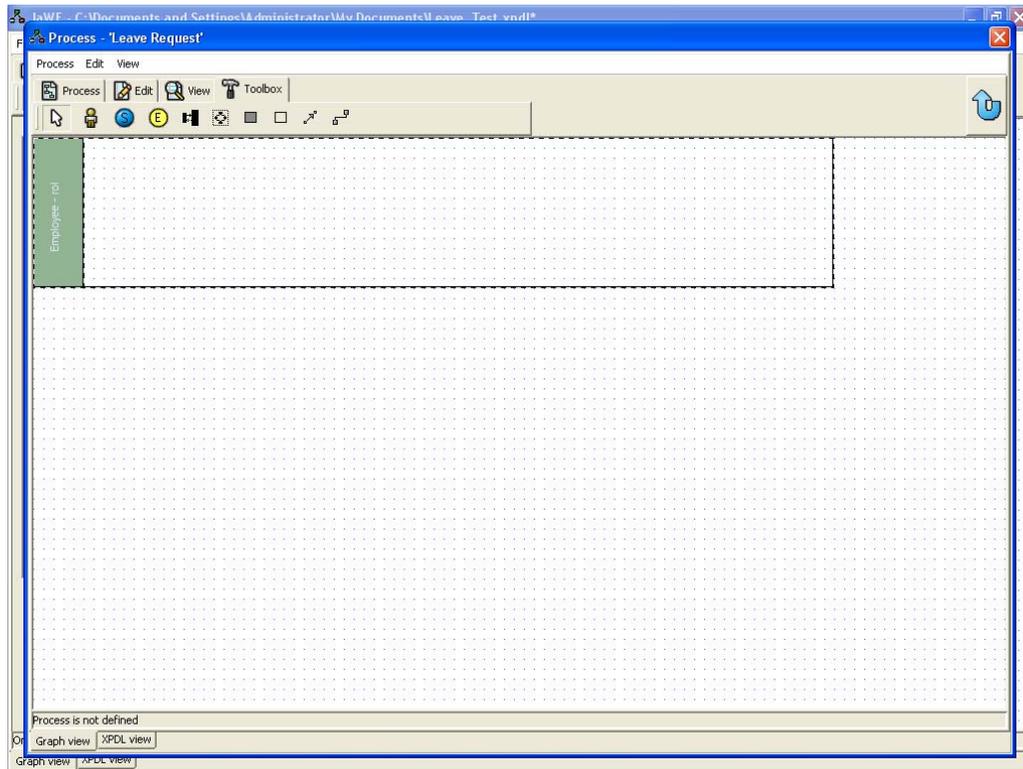
The workflow participant is defined by a type and related information that is a set of type-specific attributes. This definition contains a basic set of six workflow participant types: *resource set*, *resource*, *organizational unit*, *role*, *human*, or *system*. A role and a resource are used in the sense of abstract actors. This definition is an abstraction level between the real performer and the activity that has to be performed. During run time these abstract definitions are evaluated and assigned to concrete human(s) and/or program(s). To create roles in Zope, a user's participant type must be set to "Role".

To Create Participants:

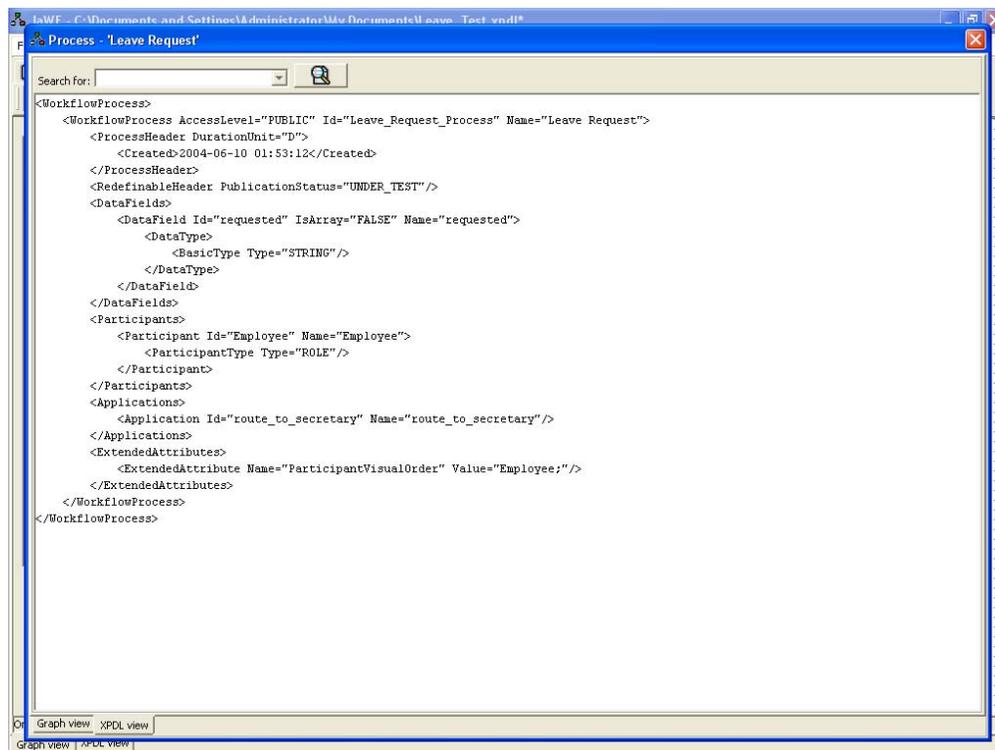
1. Click on the Participants icon at the process level. Drag and drop it on the graph editor. A new participant window will display. Complete the ID and Name fields. Select a role and enter the required properties. Click the OK button.



- The participant is created. After participants are created, activities and transitions can be added.



- The following XPDL code will be produced.



4.7 Activities

Generally, processes are comprised of a number of steps that lead towards an overall goal. Workflow processes consist of a number of workflow activities. Each workflow activity is a task that will be done by a combination of resources and computer applications.

Activities are associated with their performers (workflow participants) and application assignments. Optional information about an activity may be associated with the starting and stopping manner, usage of specific workflow relevant data, preconditions for starting, and postconditions for finishing the activity.

Generic

Most activities are atomic (generic activity), the smallest units of work, although even atomic activity may produce more than one work item for a performer, or may invoke more than one application.

SubFlow

Subflow is another activity type that implements a whole new workflow process. Process definitions within the subflow are entirely independent of the first one (where the subflow activity resides). Each subflow has its own set of activities, internal transitions, participants, application definitions, and other workflow relevant data. Participants, application definitions, and other workflow relevant data may be inherited from the model that is common for both workflow process definitions.

Block acti...

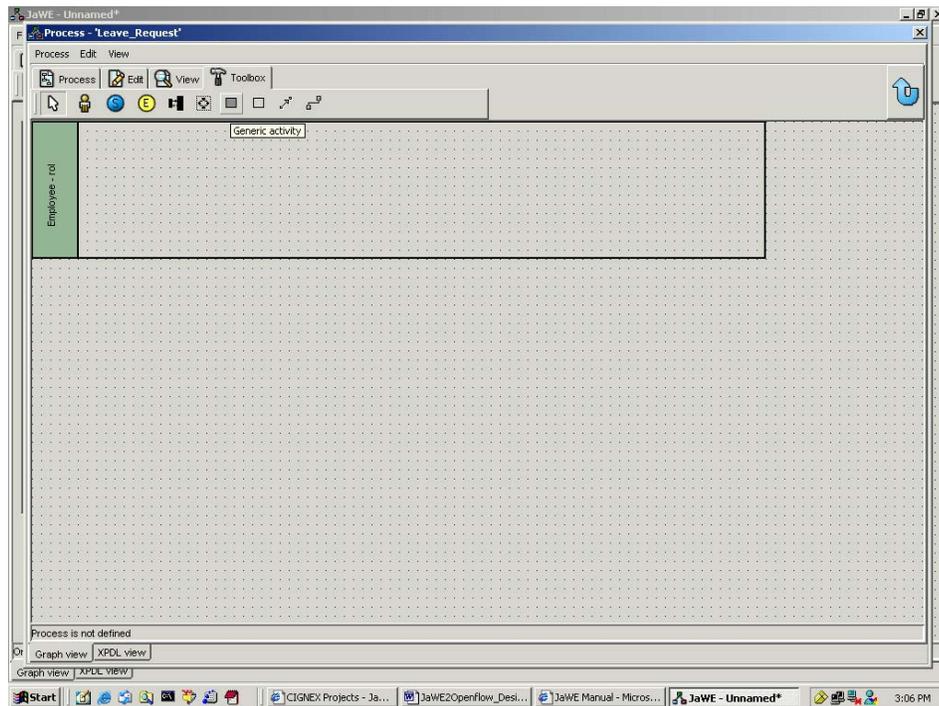
An activity may be a block activity that executes an activity set or a map of activities and transitions. Zope OpenFlow supports only independent activities and not block activities. The work-around is to create a separate process with a set of activities.

Route

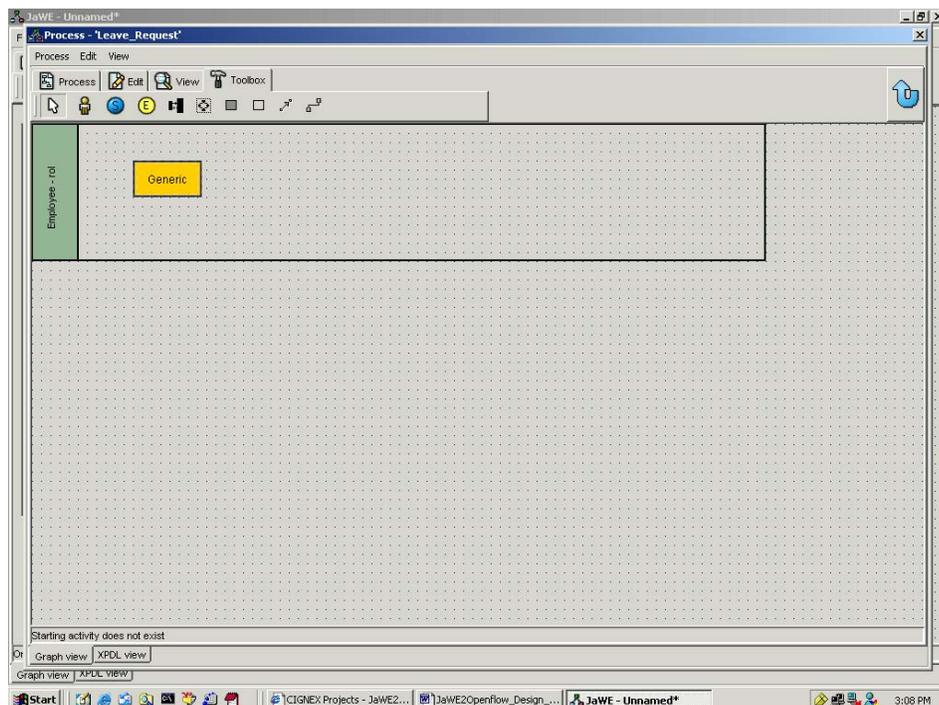
A dummy (route) activity does nothing on its own. This type of activity is used for synchronization and constructing complex and sophisticated transitional conditions (i.e., activity pre- and post- conditions).

To Create an Activity:

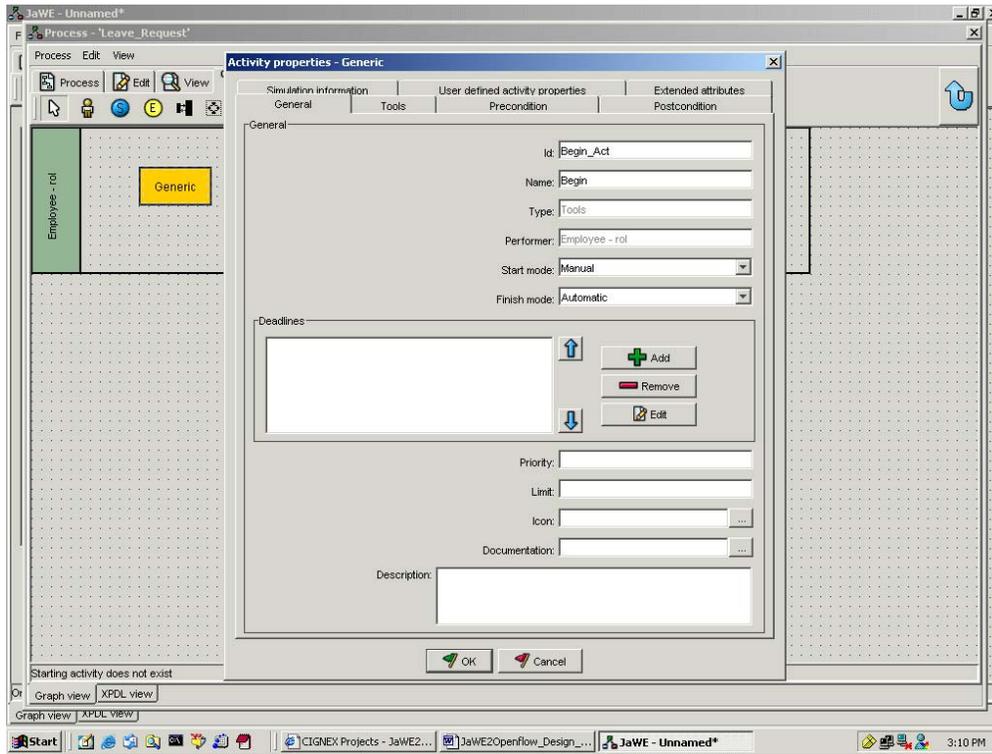
1. Click on the activity icon. Drag and drop it on the graph editor.



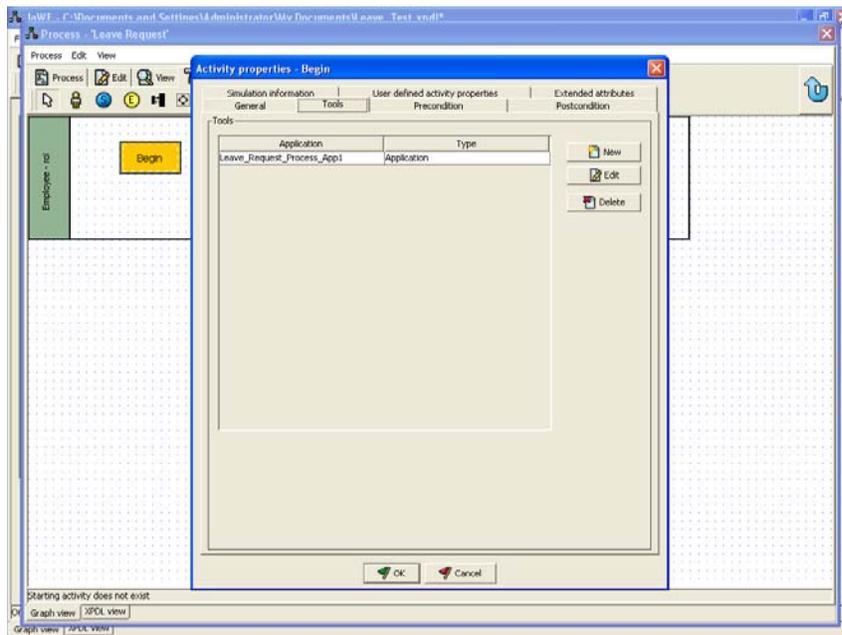
2. Double click on the activity (or right click and select Properties). The Activities Property window appears.



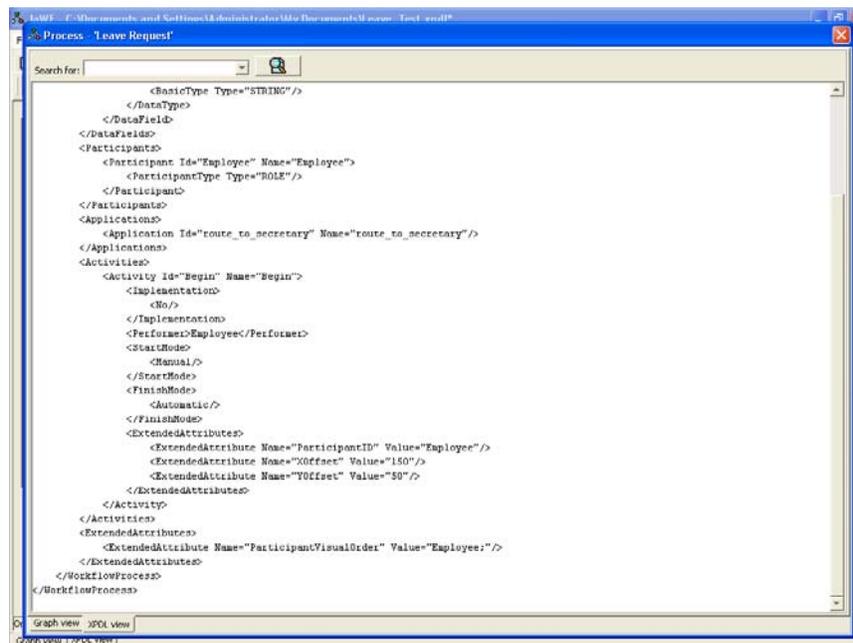
3. Select the General tab and enter the ID, Name, Start mode, and Finish mode.



4. Select the Tools tab, click on the New button and add the Applications.
5. Add the application that you want to perform in the activity. In the Type field, select “Procedure” to create a “Push Application” in the Zope OpenFlow activity. If the Type field is set to “Application”, it will be considered as “Application Name” in the Zope OpenFlow activity.
6. Click on the Pre-condition tab and select the Join type (and/xor). Click on the Post-condition tab and select Split type. Click the OK button.



7. The following XPD code will be produced.

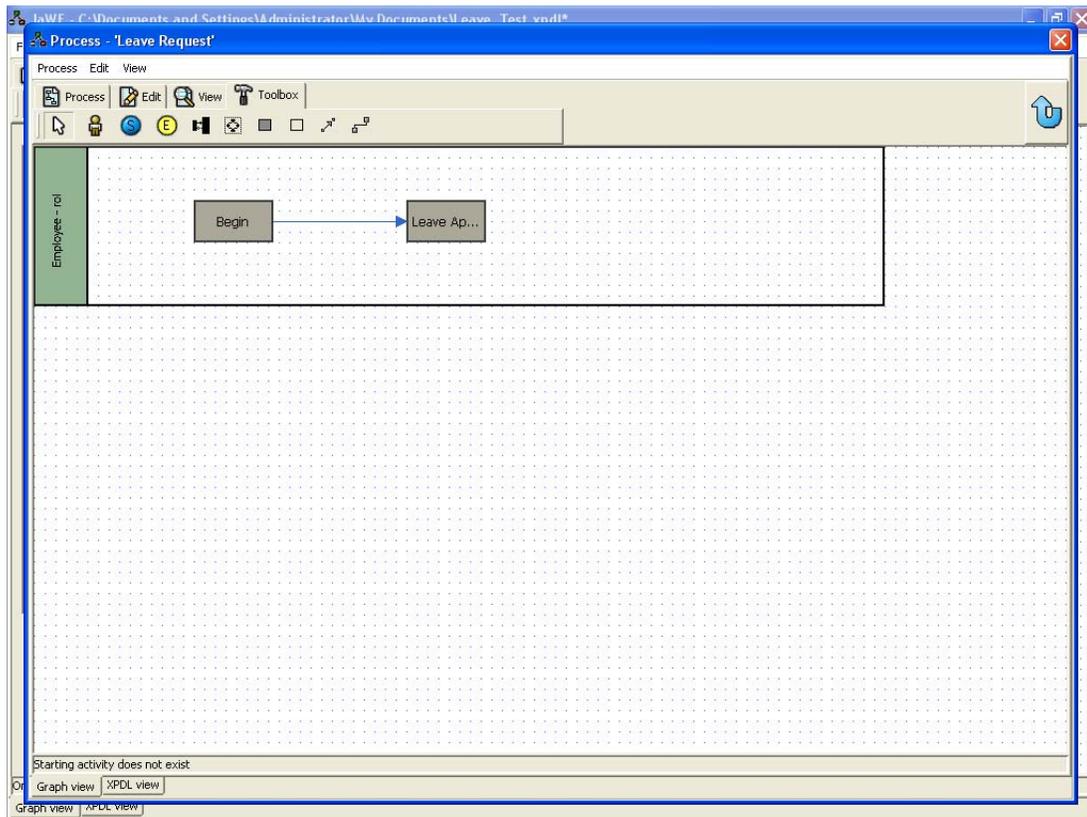


4.8 Transitions

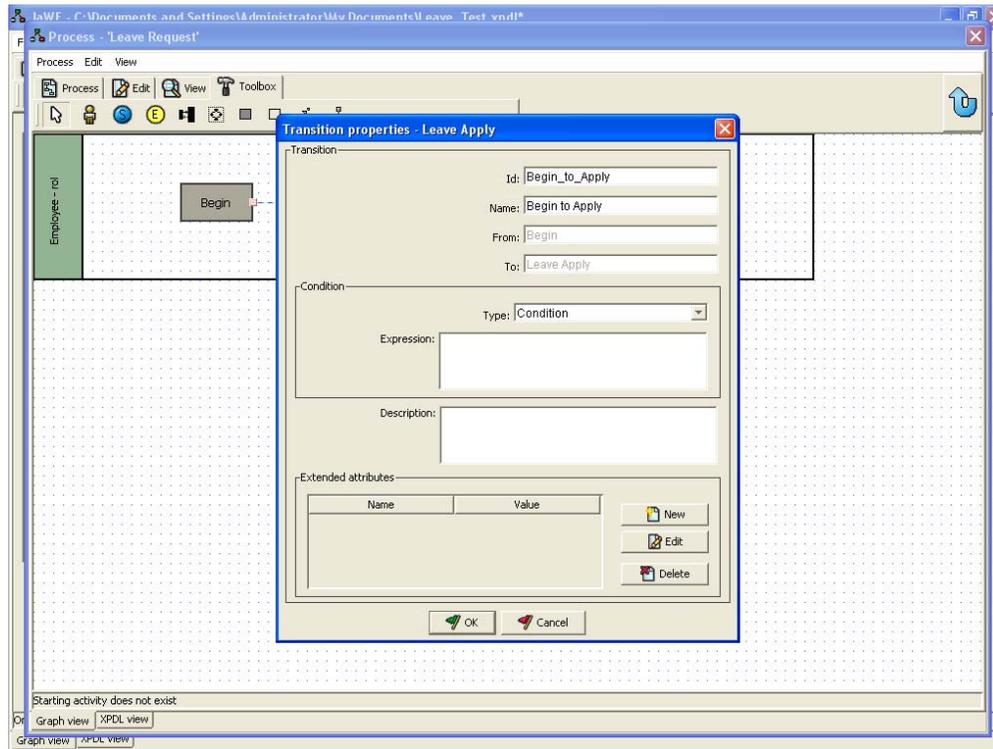
Links between two activities are established by transitions. Transitions also describe possible links between activities and the conditions that enable or disable them during workflow execution. JaWE has two (graphical) types of transitions: simple and self-routed. Simple transitions are links between two activities with one straight line. Self-routed transitions are links between two activities that are split into three parts. Icons for creating a transition are:  and .

To Create a Transition:

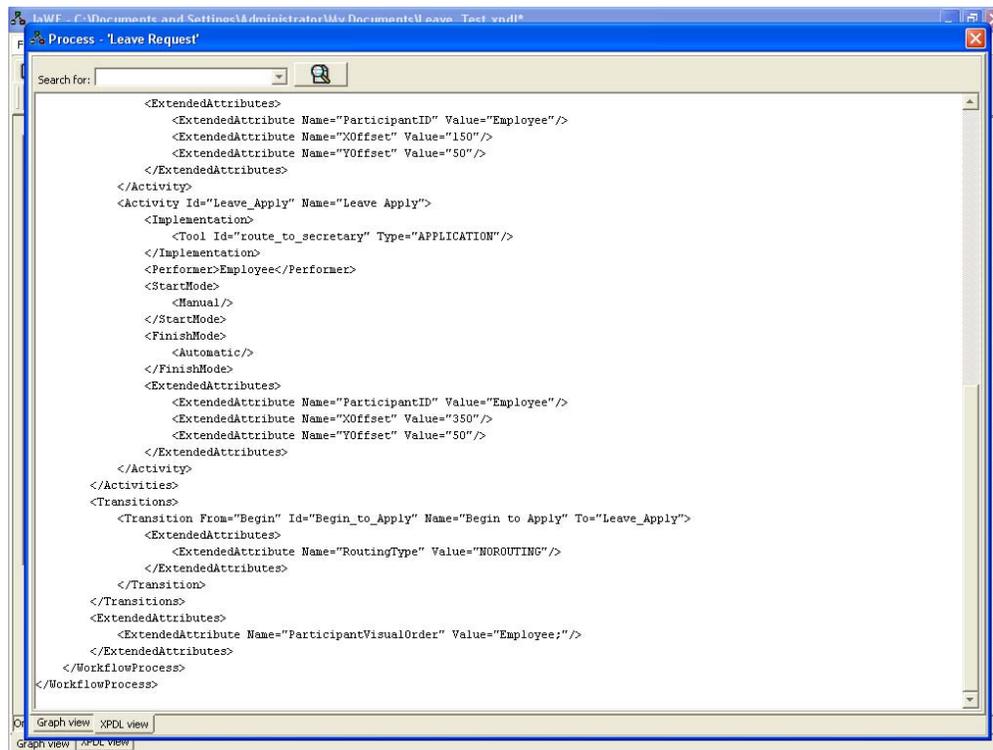
1. Click on the Transitions icon. Highlight the activity by placing the cursor on it. Drag and drop it on the other activity.



2. Double click on the transition and the Transitions Properties window appears. Complete the ID and Name fields and click OK. The transition is created.



3. The following XPD code is produced.

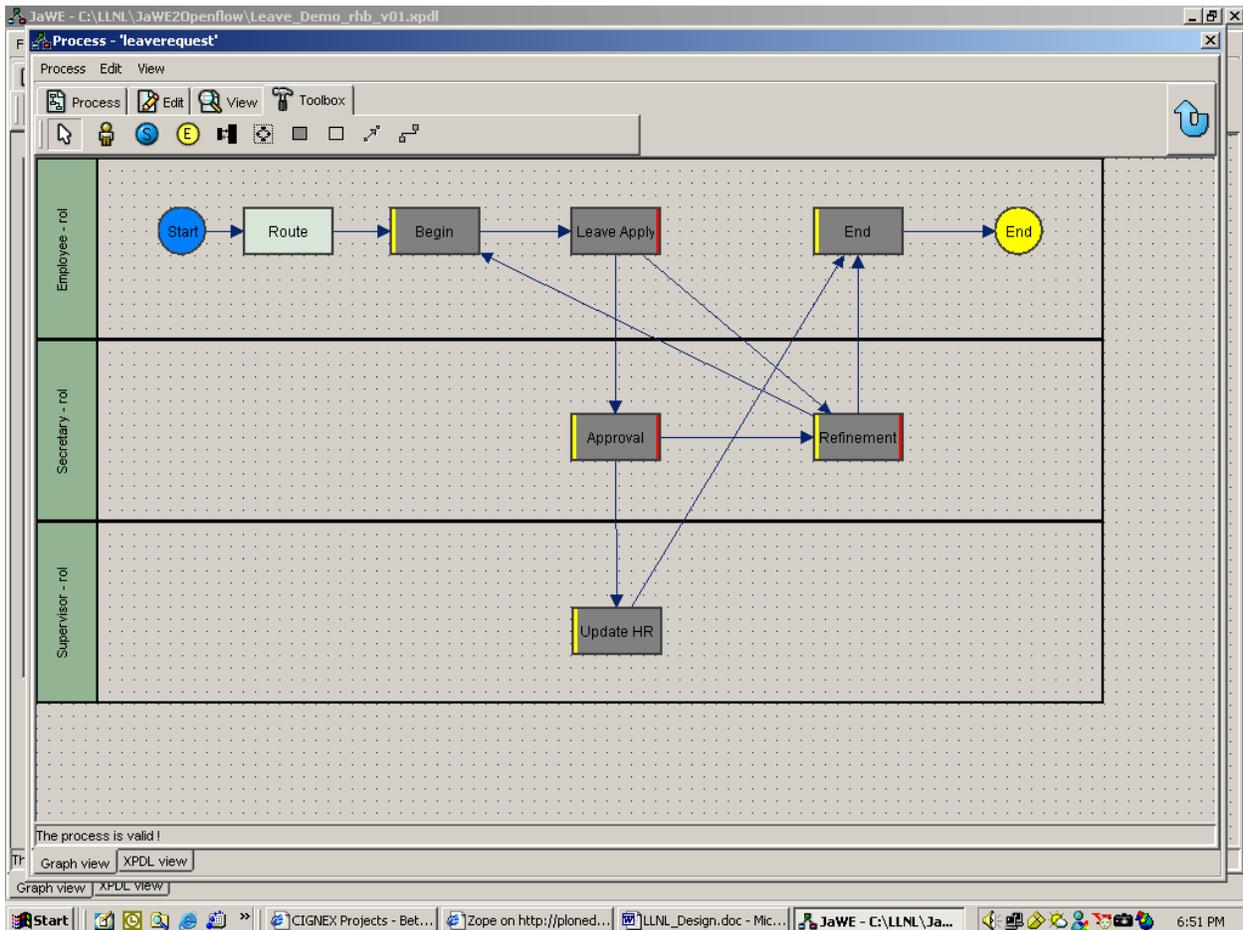


4.9 Complete JaWE Process

Below is an example workflow for leave/vacation requests.

An employee requests a leave of absence for vacation by filling out a form. The request goes to the department administrator who confirms the available vacation balance and forwards the request to a supervisor for approval. If not approved, it is returned to the employee for modification or cancellation. If approved, the request is sent to the administrator to update the HR system and notify the employee that the leave is approved.

Below is the JaWE screen for this workflow.



Below is the XPDL code output for this workflow:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package Id="Leave_Demo" xmlns="http://www.wfmc.org/2002/XPDL1.0"
xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0 http://wfmc.org/standards/docs/TC-
1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>1.0</XPDLVersion>
    <Vendor>Together</Vendor>
    <Created>2004-05-19 11:40:44</Created>
  </PackageHeader>
  <RedefinableHeader PublicationStatus="UNDER_TEST"/>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <WorkflowProcesses>
    <WorkflowProcess AccessLevel="PUBLIC" Id="leaverequest" Name="leaverequest">
      <ProcessHeader DurationUnit="D">
        <Created>2004-05-19 11:42:32</Created>
        <Priority>0</Priority>
      </ProcessHeader>
      <RedefinableHeader PublicationStatus="UNDER_TEST"/>
      <DataFields>
        <DataField Id="requested" IsArray="FALSE" Name="requested">
          <DataType>
            <BasicType Type="STRING"/>
          </DataType>
          <InitialValue>ok</InitialValue>
          <Description>Is leave requested?</Description>
        </DataField>
        <DataField Id="approved" IsArray="FALSE" Name="approved">
          <DataType>
            <BasicType Type="STRING"/>
          </DataType>
          <InitialValue>ok</InitialValue>
          <Description>Is leave approved?</Description>
        </DataField>
        <DataField Id="formalia" IsArray="FALSE" Name="formalia">
          <DataType>
            <BasicType Type="STRING"/>
          </DataType>
          <InitialValue>ok</InitialValue>
          <Description>Check that the dates are meaningful
Calculate the number of days this will take from annual leave
Check that the requester has the necessary amount of days in SIC</Description>
        </DataField>
      </DataFields>
    </WorkflowProcess>
  </WorkflowProcesses>
  <Participants>
    <Participant Id="Secretary">
      <ParticipantType Type="ROLE"/>
    </Participant>
    <Participant Id="Supervisor">
      <ParticipantType Type="ROLE"/>
    </Participant>
    <Participant Id="Employee">
      <ParticipantType Type="ROLE"/>
    </Participant>
  </Participants>
</Package>
```

```

</Participants>
<Applications>
  <Application Id="leave_refine"/>
  <Application Id="leave_checkstatus"/>
  <Application Id="leave_approvalform"/>
  <Application Id="route_to_secretary"/>
  <Application Id="route_to_customer"/>
  <Application Id="leave_hrform"/>
  <Application Id="leave_finalinfo"/>
  <Application Id="route_to_supervisor"/>
</Applications>
<Activities>
  <Activity Id="Approval" Name="Approval">
    <Implementation>
      <Tool Id="leave_approvalform" Type="APPLICATION">
        <Description>This is leave approval form</Description>
      </Tool>
      <Tool Id="route_to_supervisor" Type="PROCEDURE"/>
    </Implementation>
    <Performer>Secretary</Performer>
    <StartMode>
      <Manual/>
    </StartMode>
    <FinishMode>
      <Automatic/>
    </FinishMode>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Split Type="XOR">
          <TransitionRefs>
            <TransitionRef Id="request_approved"/>
            <TransitionRef Id="not_approved"/>
          </TransitionRefs>
        </Split>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes>
      <ExtendedAttribute Name="ParticipantID" Value="Secretary"/>
      <ExtendedAttribute Name="XOffset" Value="440"/>
      <ExtendedAttribute Name="YOffset" Value="60"/>
    </ExtendedAttributes>
  </Activity>
  <Activity Id="UpdateHR" Name="Update HR">
    <Implementation>
      <Tool Id="leave_hrform" Type="APPLICATION"/>
      <Tool Id="route_to_secretary" Type="PROCEDURE"/>
    </Implementation>
    <Performer>Supervisor</Performer>
    <StartMode>
      <Manual/>
    </StartMode>
    <FinishMode>
      <Automatic/>
    </FinishMode>
    <ExtendedAttributes>
      <ExtendedAttribute Name="ParticipantID" Value="Supervisor"/>

```

```

    <ExtendedAttribute Name="XOffset" Value="441"/>
    <ExtendedAttribute Name="YOffset" Value="70"/>
  </ExtendedAttributes>
</Activity>
<Activity Id="Refinement" Name="Refinement">
  <Implementation>
    <Tool Id="leave_refine" Type="APPLICATION"/>
    <Tool Id="route_to_customer" Type="PROCEDURE"/>
  </Implementation>
  <Performer>Secretary</Performer>
  <StartMode>
    <Manual/>
  </StartMode>
  <FinishMode>
    <Automatic/>
  </FinishMode>
  <TransitionRestrictions>
    <TransitionRestriction>
      <Join Type="XOR"/>
      <Split Type="XOR">
        <TransitionRefs>
          <TransitionRef Id="cancel_request"/>
          <TransitionRef Id="refinement_begin"/>
        </TransitionRefs>
      </Split>
    </TransitionRestriction>
  </TransitionRestrictions>
  <ExtendedAttributes>
    <ExtendedAttribute Name="ParticipantID" Value="Secretary"/>
    <ExtendedAttribute Name="XOffset" Value="640"/>
    <ExtendedAttribute Name="YOffset" Value="60"/>
  </ExtendedAttributes>
</Activity>
<Activity Id="Leave_Apply" Name="Leave Apply">
  <Implementation>
    <Tool Id="route_to_secretary" Type="PROCEDURE"/>
    <Tool Id="leave_checkstatus" Type="APPLICATION"/>
  </Implementation>
  <Performer>Employee</Performer>
  <StartMode>
    <Manual/>
  </StartMode>
  <FinishMode>
    <Automatic/>
  </FinishMode>
  <TransitionRestrictions>
    <TransitionRestriction>
      <Split Type="XOR">
        <TransitionRefs>
          <TransitionRef Id="send_to_approval"/>
          <TransitionRef Id="send_to_refinement"/>
        </TransitionRefs>
      </Split>
    </TransitionRestriction>
  </TransitionRestrictions>
  <ExtendedAttributes>

```

```

        <ExtendedAttribute Name="ParticipantID" Value="Employee"/>
        <ExtendedAttribute Name="XOffset" Value="440"/>
        <ExtendedAttribute Name="YOffset" Value="40"/>
    </ExtendedAttributes>
</Activity>
<Activity Id="Begin" Name="Begin">
    <Implementation>
        <No/>
    </Implementation>
    <Performer>Employee</Performer>
    <StartMode>
        <Manual/>
    </StartMode>
    <FinishMode>
        <Automatic/>
    </FinishMode>
    <TransitionRestrictions>
        <TransitionRestriction>
            <Join Type="XOR"/>
        </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes>
        <ExtendedAttribute Name="ParticipantID" Value="Employee"/>
        <ExtendedAttribute Name="XOffset" Value="291"/>
        <ExtendedAttribute Name="YOffset" Value="40"/>
    </ExtendedAttributes>
</Activity>
<Activity Id="End" Name="End">
    <Implementation>
        <Tool Id="leave_finalinfo" Type="APPLICATION"/>
        <Tool Id="route_to_customer" Type="PROCEDURE"/>
    </Implementation>
    <Performer>Employee</Performer>
    <StartMode>
        <Manual/>
    </StartMode>
    <FinishMode>
        <Manual/>
    </FinishMode>
    <TransitionRestrictions>
        <TransitionRestriction>
            <Join Type="XOR"/>
        </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes>
        <ExtendedAttribute Name="ParticipantID" Value="Employee"/>
        <ExtendedAttribute Name="XOffset" Value="640"/>
        <ExtendedAttribute Name="YOffset" Value="40"/>
    </ExtendedAttributes>
</Activity>
<Activity Id="leaverequest_Act1" Name="Route">
    <Route/>
    <StartMode>
        <Automatic/>
    </StartMode>
    <FinishMode>

```

```

    <Automatic/>
  </FinishMode>
  <ExtendedAttributes>
    <ExtendedAttribute Name="ParticipantID" Value="Employee"/>
    <ExtendedAttribute Name="XOffset" Value="170"/>
    <ExtendedAttribute Name="YOffset" Value="40"/>
  </ExtendedAttributes>
</Activity>
</Activities>
<Transitions>
  <Transition From="Approval" Id="request_approved" Name="request_approved" To="UpdateHR">
    <Condition Type="CONDITION">python:instance.approved == 'ok'</Condition>
    <ExtendedAttributes>
      <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
    </ExtendedAttributes>
  </Transition>
  <Transition From="Approval" Id="not_approved" Name="not_approved" To="Refinement">
    <Condition Type="CONDITION">python:instance.approved != 'ok'</Condition>
    <ExtendedAttributes>
      <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
    </ExtendedAttributes>
  </Transition>
  <Transition From="Leave_Apply" Id="send_to_refinement" Name="Transition" To="Refinement">
    <Condition Type="CONDITION">python:instance.formalia != 'ok'</Condition>
    <ExtendedAttributes>
      <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
    </ExtendedAttributes>
  </Transition>
  <Transition From="Leave_Apply" Id="send_to_approval" Name="Transition" To="Approval">
    <Condition Type="CONDITION">python:instance.formalia == 'ok'</Condition>
    <ExtendedAttributes>
      <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
    </ExtendedAttributes>
  </Transition>
  <Transition From="Begin" Id="begin_to_apply" Name="Transition" To="Leave_Apply">
    <ExtendedAttributes>
      <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
    </ExtendedAttributes>
  </Transition>
  <Transition From="Refinement" Id="cancel_request" Name="Transition" To="End">
    <Condition Type="CONDITION">python:instance.requested != 'ok'</Condition>
    <ExtendedAttributes>
      <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
    </ExtendedAttributes>
  </Transition>
  <Transition From="Refinement" Id="refinement_begin" Name="Transition" To="Begin">
    <Condition Type="CONDITION">python:instance.requested == 'ok'</Condition>
    <ExtendedAttributes>
      <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
    </ExtendedAttributes>
  </Transition>
  <Transition From="UpdateHR" Id="tell_employee" Name="Transition" To="End">
    <ExtendedAttributes>
      <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
    </ExtendedAttributes>
  </Transition>

```

```
<Transition From="leaverequest_Act1" Id="leaverequest_Tra10" Name="Transition" To="Begin">
  <ExtendedAttributes>
    <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
  </ExtendedAttributes>
</Transition>
</Transitions>
<ExtendedAttributes>
  <ExtendedAttribute Name="StartOfWorkflow"
Value="Employee;leaverequest_Act1;100;40;NOROUTING"/>
  <ExtendedAttribute Name="EndOfWorkflow" Value="Employee;End;790;40;NOROUTING"/>
  <ExtendedAttribute Name="ParticipantVisualOrder" Value="Employee;Secretary;Supervisor;"/>
</ExtendedAttributes>
</WorkflowProcess>
</WorkflowProcesses>
<ExtendedAttributes>
  <ExtendedAttribute Name="MadeBy" Value="JaWE"/>
  <ExtendedAttribute Name="Version" Value="1.2"/>
</ExtendedAttributes>
</Package>
```

5 JaWE2Openflow Product

5.1 Prerequisites

The JaWE2Openflow product has following prerequisites:

- Zope 2.6.0 or higher.
- OpenFlow 1.2.0 product.
- Python XML library – PyXML-0.8.3.tgz.

This product can be installed in Plone 2.0 or Zope 2.6.0 or higher/CMF 1.4 websites.

5.2 How to Install

Installing JaWE2Openflow is just like installing any other Zope product:

To Install JaWE2Openflow:

1. Install Zope 2.6.0 or higher (see <http://www.zope.org> for download and instructions).
2. Install OpenFlow 1.2.0 product in the <zope_installation_dir>/lib/python/Products folder or in the <zope_instance>/Products folder for Zope 2.7.0 (see http://www.openflow.it/Download/index_html for download and instructions).
3. Install Python XML Library - PyXML 0.8.3.tgz. This should be installed in site-packages folder. For example <Zope or Plone installation dir>/lib/python/site-packages. (Note: Search for site-packages on the system.)
4. Unpack the JaWE2Openflow.tgz in the <zope_installation_dir>/lib/python/Products folder or in the <zope_instance>/Products folder for Zope 2.7.0. The object will be present in select type to add list of Zope objects.
5. Restart Zope.

If Zope has already been installed, start from step number 2.

5.3 How to get Help

Click on the Help link in the top right corner in the ZMI to get the help on the JaWE2Openflow product. A new help window appears on the screen.

5.4 How to Use

5.4.1 Login to ZMI

Login to ZMI and go to the folder where the OpenFlow application should be created.

5.4.2 Create an Instance of the Product

Select JaWE2Openflow from the list of select types to add to ZMI and click on Add button. A screen with two input fields displays: the ID field has a default value “jawe_of” and the Title field defaults to “JaWE to OpenFlow Converter”. These default values can be used or changed as appropriate. Click the Add JaWE2Openflow button. An instance of the product that converts XPDL files to Zope OpenFlow is created.

5.4.3 Upload an XPDL file

Click on the object created in ZMI and select the Upload tab. Enter the appropriate value in the ID field and click on the Browse button. Select the XPDL file to be uploaded or type the relative path and filename. Click on the Submit button.

A folder will be created with the folder ID as <given_ID>_folder. An OpenFlow object will be created inside this folder with the specified ID. The XPDL file will also be uploaded to the folder as a reference and a message log file will be created in the folder.

A message appears with a link to the message log file. Click on the link to see the sequence of operations carried out and the warnings and errors that were encountered while converting the XPDL file to an OpenFlow object.

Select the folder for the JaWE2Openflow object. Click on the folder for the new object to see a list of objects created along with the OpenFlow object.

An “acl_users” folder is created with the users matched with the participants in the XPDL file. Users will be added only for Participants of type ‘Role’.

An “index_html” page template is created for the data fields whose data type is “BasicType” and defined at package level in the JaWE XPDL file. Modify “index_html” according to workflow requirements.

Page Templates are created for all the applications.

Click on the OpenFlow object to see a screen with the Worklist tab highlighted with all of the processes and activities involved in each process. Click on the Roles tab to see the list of all roles that can be assigned to users. Click on the Applications tab, to see all the applications listed that are present in the workflow. The Process Definitions tab shows all processes that are present. Click on a process to view a list of all activities and transitions that are present in the process. Activities can be added or deleted from this list. Click on the activity or transition to see the Edit screen where properties of the activity or transition can be edited. The Contents tab displays the list of contents. The Security tab displays the permissions for each role and allows users to assign permissions, revoke them and create new roles.

5.5 Check Output Log File

Check the output log file that is created while converting the XPDL file to OpenFlow for any further information required.

If the log file shows an error, check the XPDL file generated from JaWE. Delete the folder and upload the modified XPDL file using product instance Upload tab.

On successful conversion of JaWE XPDL to Zope OpenFlow, modify process/activity/forms and add the necessary forms/scripts to complete and use workflow application.

5.6 How to Maintain the Product

JaWE2Openflow is a Zope based product. The JaWE2Openflow product can be installed on the following flavors of Zope:

- Zope 2.7 instance
- CMF 1.4 instance
- Plone 2.0 instance

5.6.1 JaWE2Openflow Product-Related Files

Once the product is installed, go through the following files for more details:

DEPENDENCIES.txt	This file has list of required software to use JaWE2Openflow product.
Help	This is a help folder. In this folder, there is “overview.html” help file, which can be seen using the Help link in ZMI.
__init__.py	Product initialization file
INSTALL.txt	Product installation instructions
JaWE2Openflow.py	This is actual product file. All functions related to the product are defined in this class. Details of the functions are given in this class.
README.txt	Readme file
RELEASE_NOTES.txt	Release note with a list of bugs fixed in the final release.
sampleFiles	In this folder, a few sample XPDL files are given for testing purpose only.
VERSION.txt	Product version file
www	This folder has the product logo.
Zpt	This folder has all page templates used by JaWE2Openflow product.

If any changes are made to the product file:

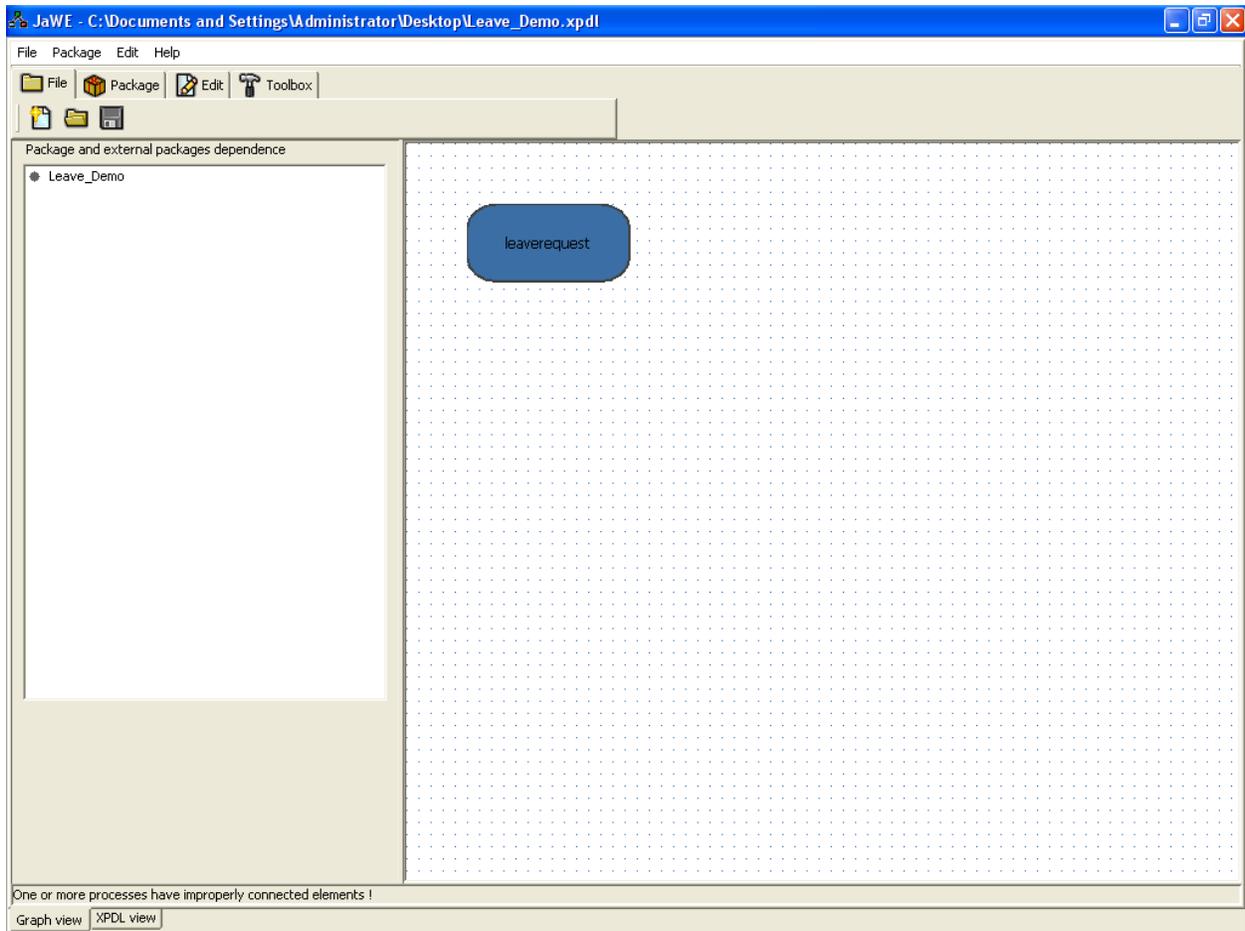
1. Restart the server.
2. Delete the existing product instance (if any) in ZMI, and create a new instance of the product by selecting “JaWE2Openflow” product from the Select Type to Add product list in ZMI.

6 JaWE to OpenFlow Conversion

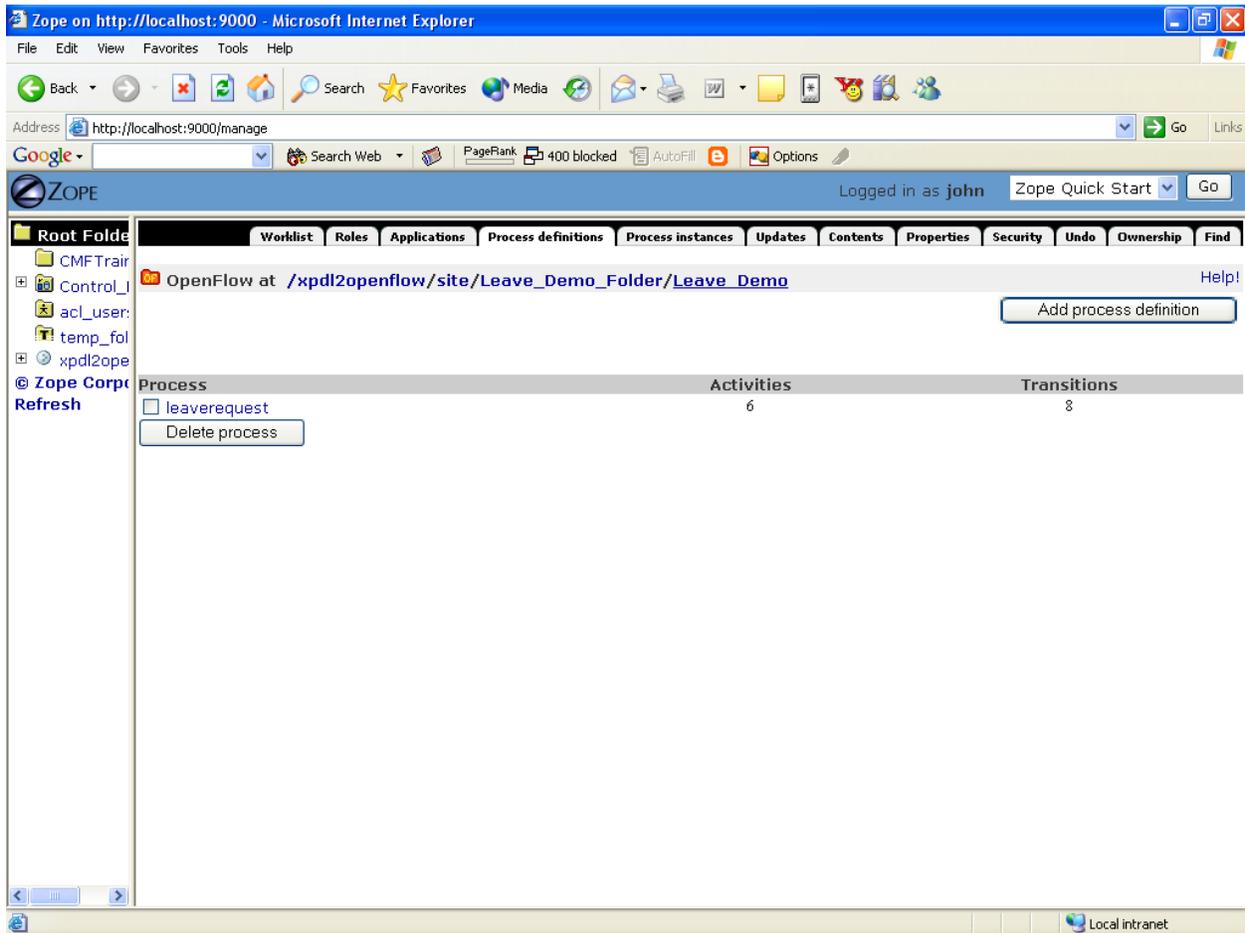
This section shows screenshots of the JAWE to OpenFlow conversion.

6.1 Process

A process is added in the JaWE editor.

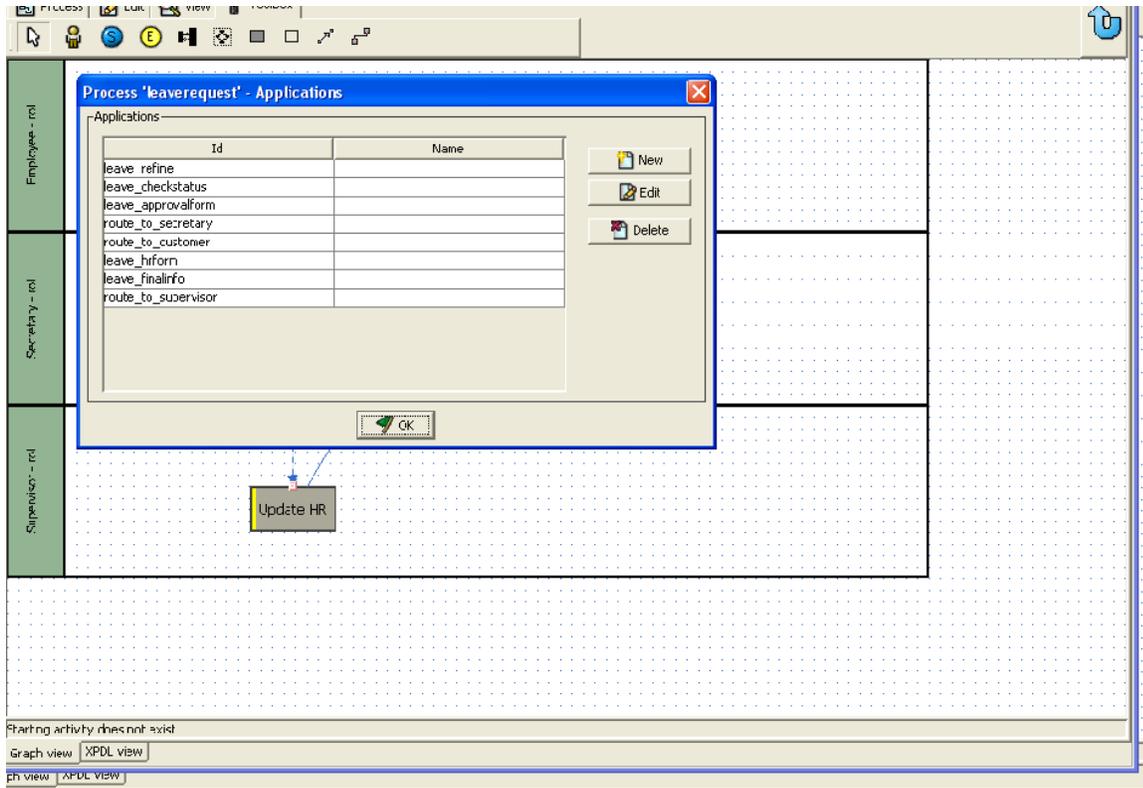


When converted to OpenFlow, the process displays on the Process definitions tab.

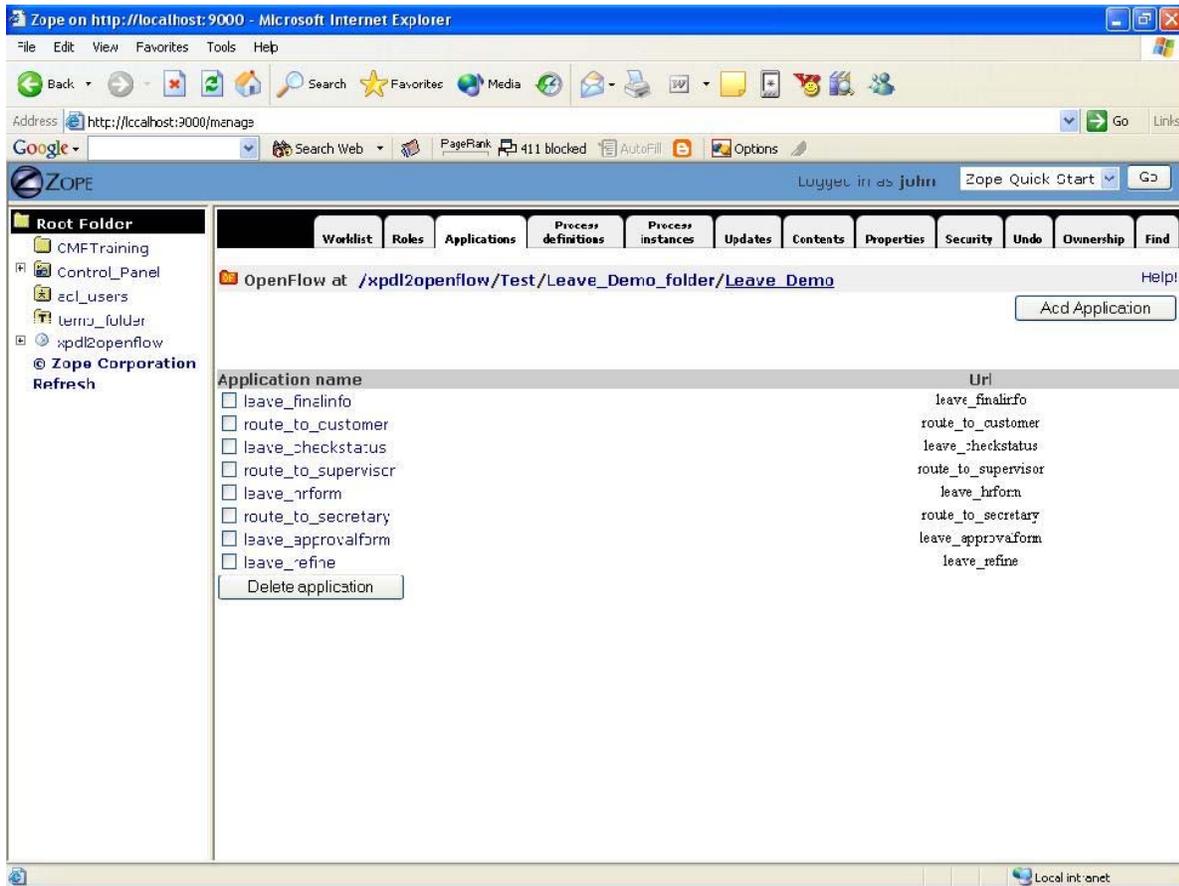


6.2 Applications

Applications are added to the process in the JaWE editor.

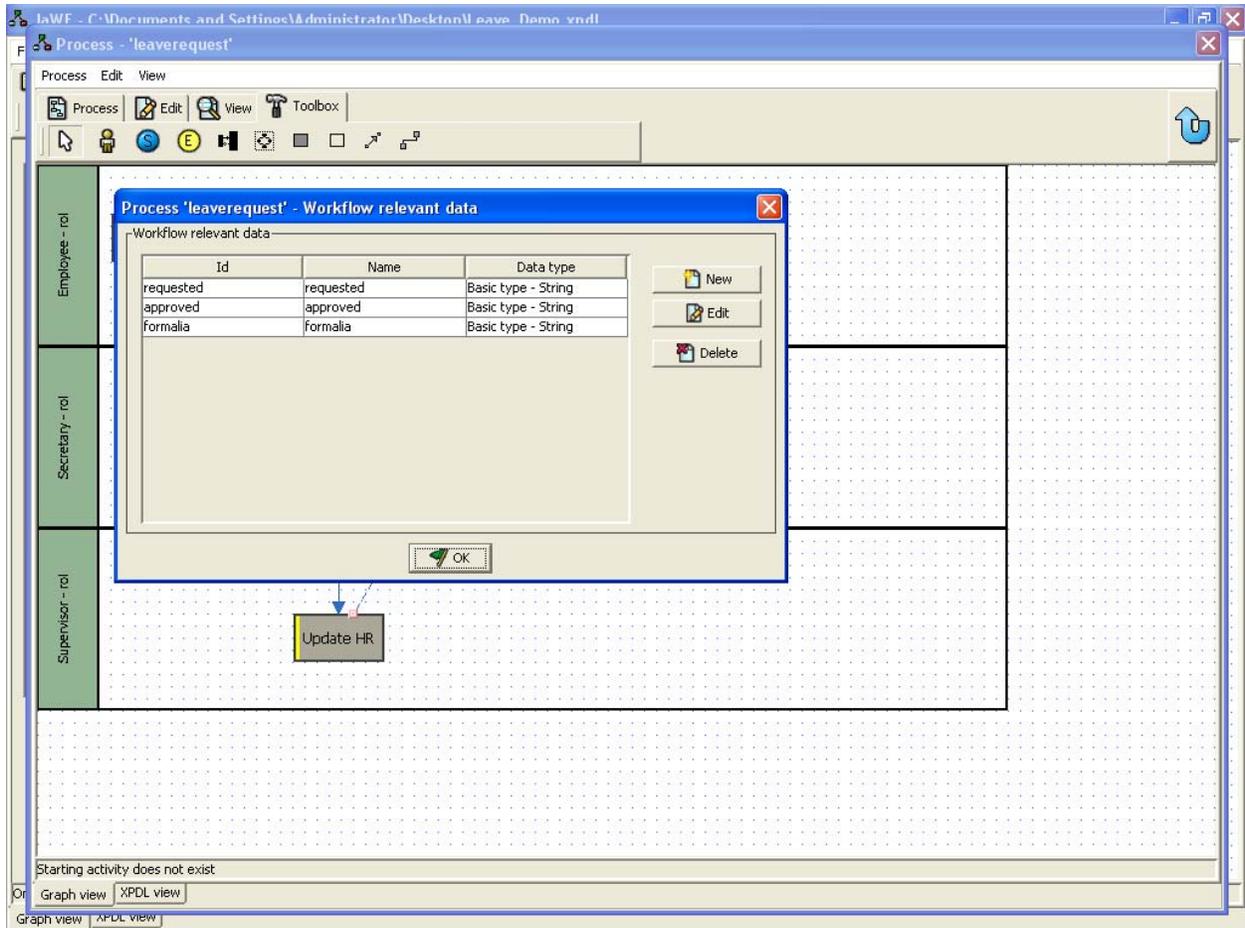


When converted to OpenFlow, these applications are listed in the Applications tab.



6.3 Workflow Relevant Data

Workflow relevant data is added to the process in the JaWE editor.

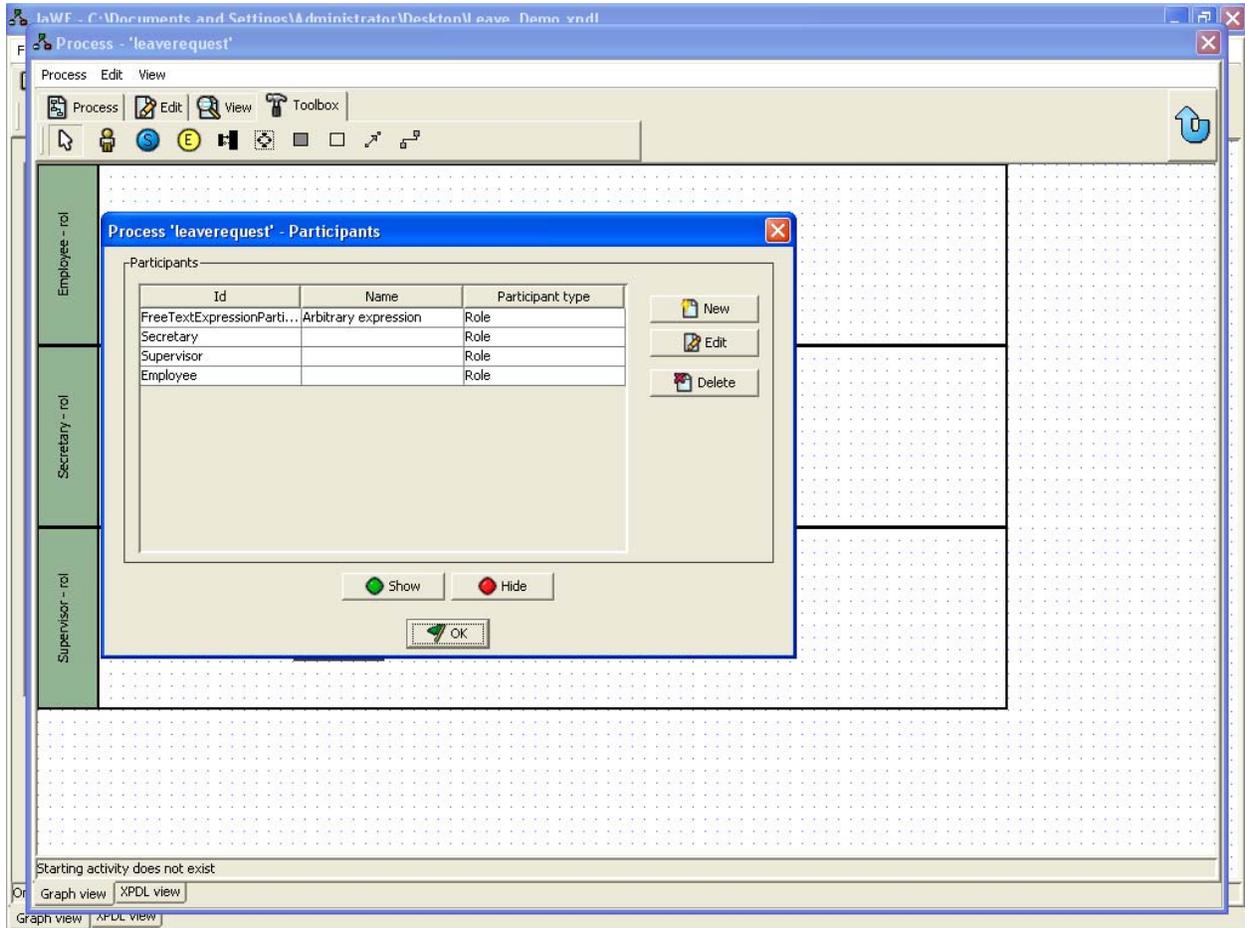


A form is created to edit workflow relevant data in the JaWE2Openflow for the datafields present in the process.

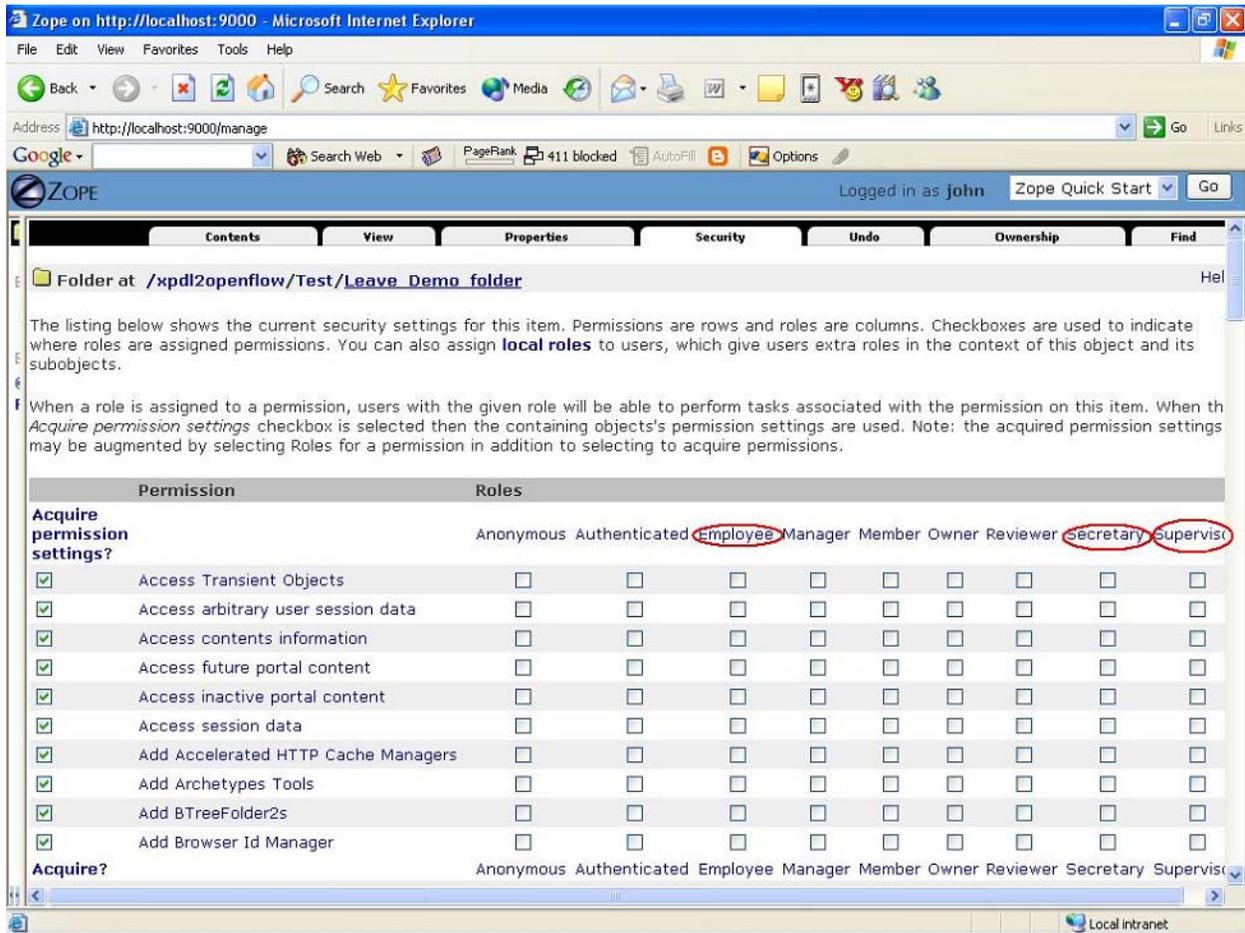
Microsoft Internet Explorer window showing the Zope management interface. The browser title is "Zope on http://localhost:9000 - Microsoft Internet Explorer". The address bar shows "http://localhost:9000/manage". The Zope interface includes a navigation menu on the left with folders like "Root Folder", "CMFTraining", "Control_Panel", "acl_users", "temp_folder", and "xpdlopenflow". The main content area displays the "Plone" logo and a "Home Page" form with fields for "requested:", "approved:", and "formalia:". A calendar for June 2004 is visible on the right, with the 9th highlighted. The footer contains copyright information and standards compliance links.

6.4 Participants

Participants are added to the process in the JaWE editor.

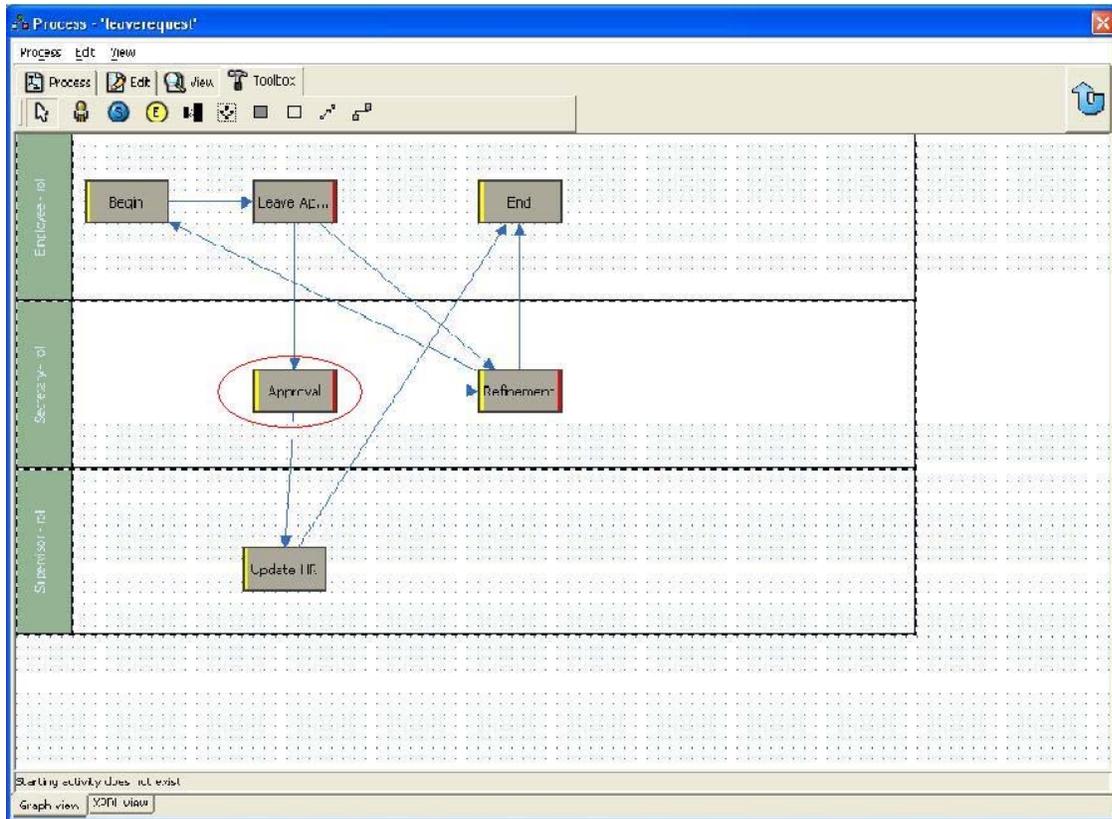


In JaWE2Openflow, for the participants with the type “role” a role is created at the security tab at folder level.

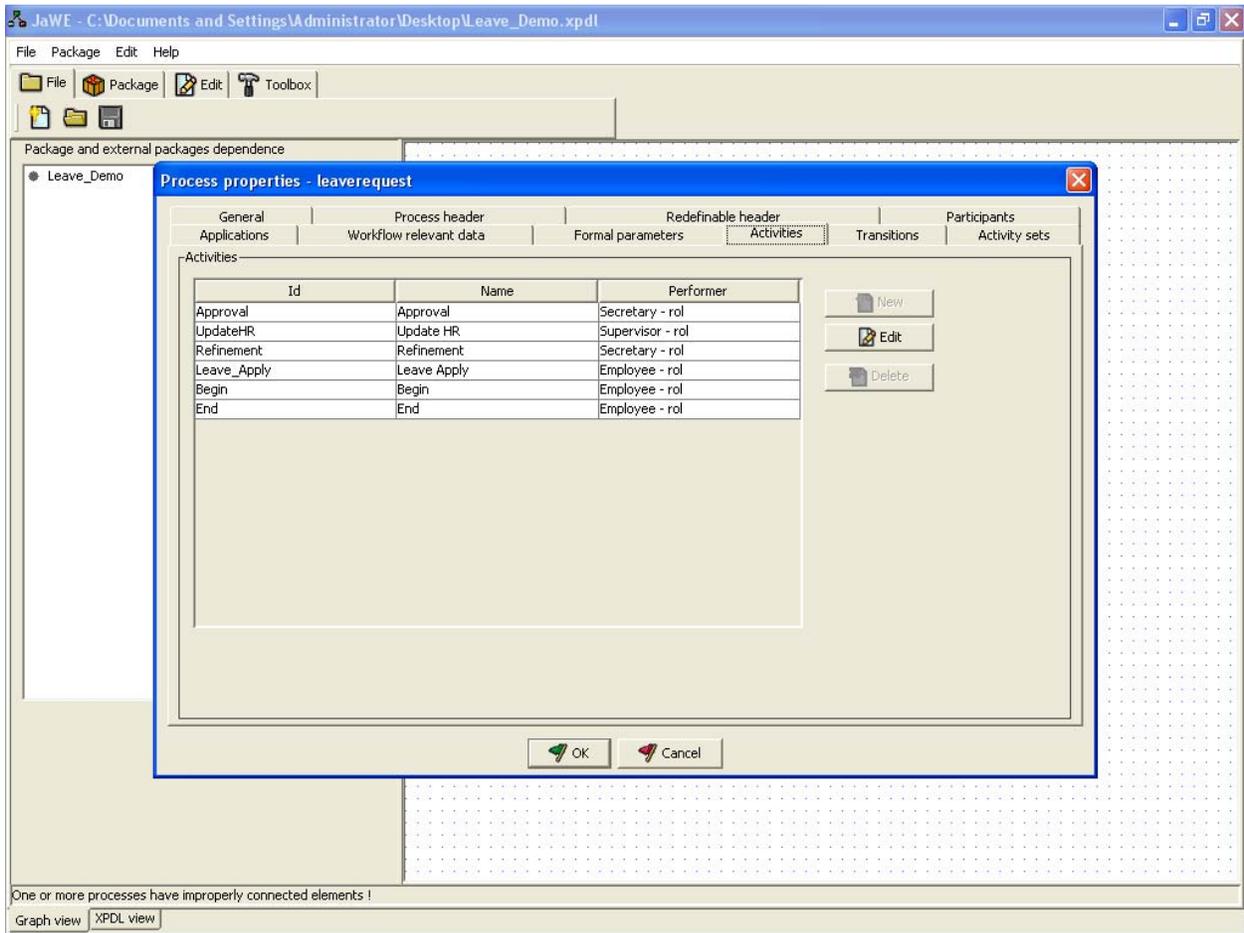


6.5 Activities

Activities are added to the process in the JaWE editor.



The list of activities at the Process-Properties level shown below was created in JaWE2Openflow.



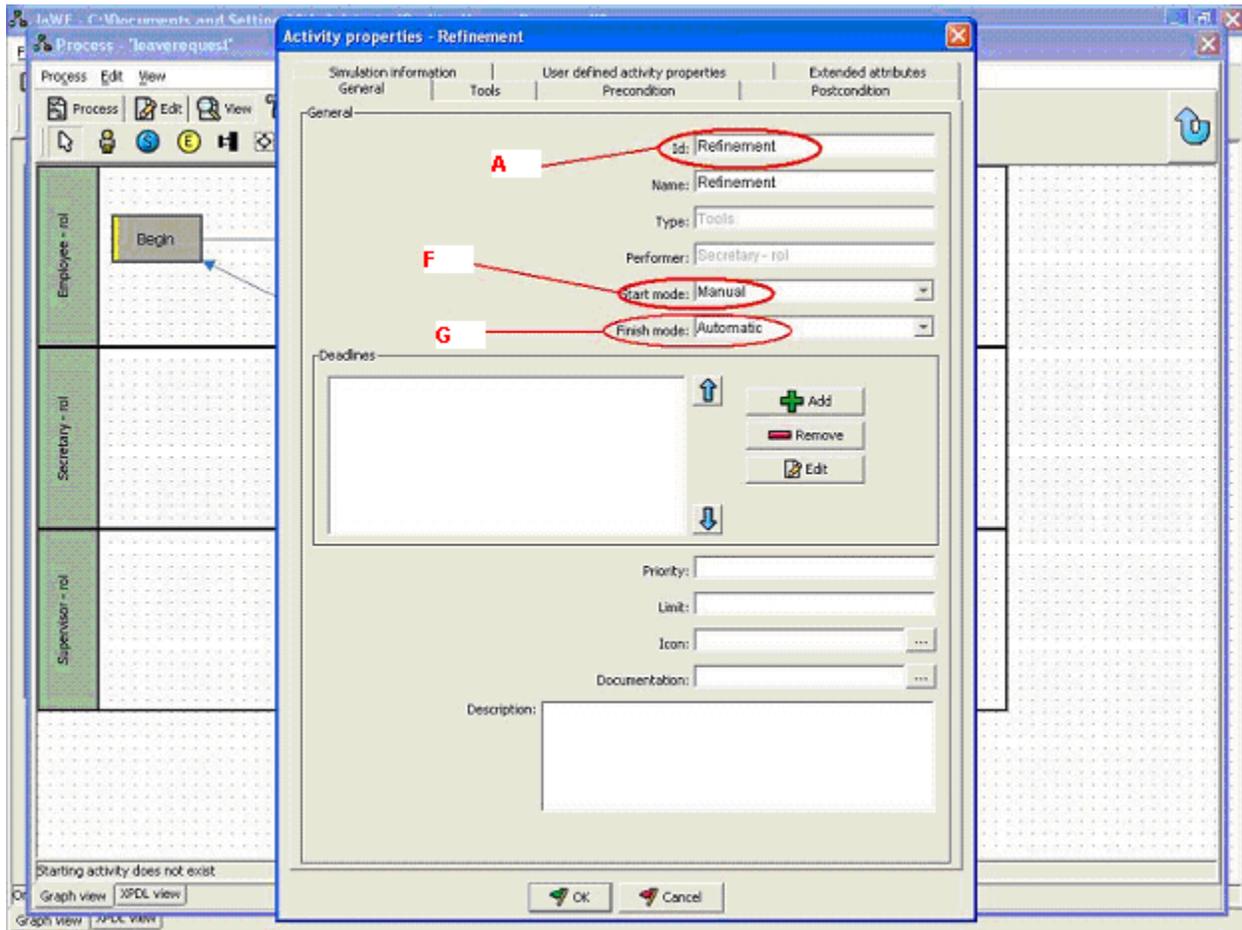
When converted by JaWE2Openflow, the activities will display on the Map tab. The screen shots on the following pages identify the marked sections of this graphic (A-G).

The screenshot shows the ZOPE web interface in Microsoft Internet Explorer. The browser address is `http://localhost:9000/manage`. The user is logged in as 'john'. The interface displays the 'Map' tab for a process at `/xpd2openflow/site/AAA111_Folder/AAA111/leaverequest`. The 'Activities' table lists various process steps, and the 'Transitions' table lists the conditions and flow between them. Red annotations highlight specific parts: 'A' points to the 'Refinement' activity; 'B' points to the 'Join' column for 'Refinement'; 'C' points to the 'Split' column for 'Refinement'; 'D' points to the 'Application name' 'leave_refine'; 'E' points to the 'Push Application' 'route_to_customer'; 'F' points to the 'Start mode' 'Manual'; and 'G' points to the 'Finish mode' 'Automatic'.

Activity	Kind	Join	Split	Application name	Push Application	Start mode	Finish mode	Subflow process
<input type="checkbox"/> Approval	standard	and	xor	leave_approvalform	route_to_supervisor	Manual	Automatic	
<input type="checkbox"/> UpdateHR	standard	and	and	leave_hrform	route_to_secretary	Manual	Automatic	
<input type="checkbox"/> Refinement	standard	xor	xor	leave_refine	route_to_customer	Manual	Automatic	
<input type="checkbox"/> Leave_Apply	standard	and	xor	leave_checkstatus	route_to_secretary	Manual	Automatic	
<input type="checkbox"/> Begin	standard	and	and			Manual	Automatic	
<input type="checkbox"/> End	standard	xor	and	leave_finalinfo	route_to_customer	Manual	Manual	

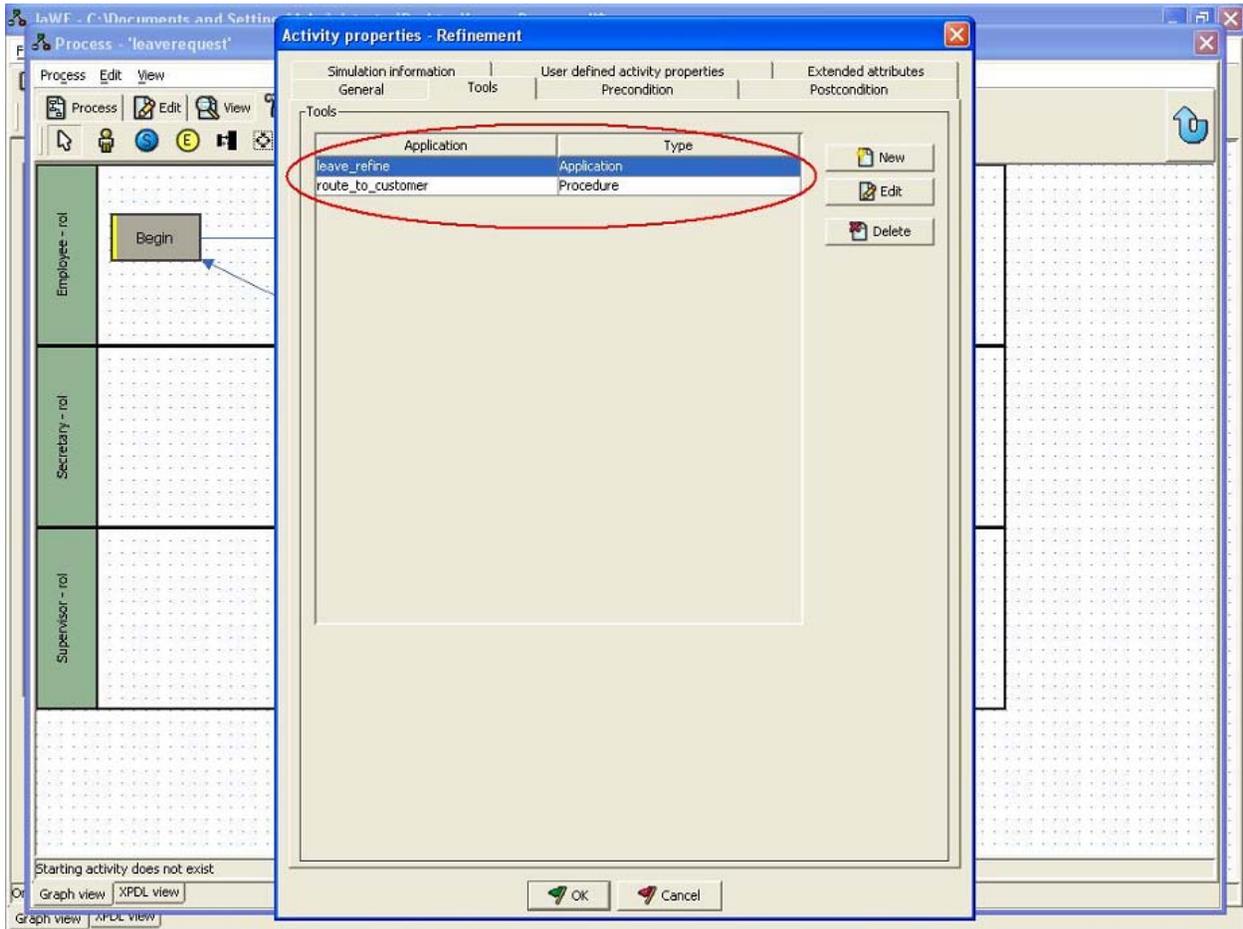
Transition	Condition	From	To
<input type="checkbox"/> request_approved	python:instance.approved == 'ok'	Approval	UpdateHR
<input type="checkbox"/> not_approved	python:instance.approved != 'ok'	Approval	Refinement
<input type="checkbox"/> send_to_refinement	python:instance.formalia != 'ok'	Leave_Apply	Refinement
<input type="checkbox"/> send_to_approval	python:instance.formalia == 'ok'	Leave_Apply	Approval
<input type="checkbox"/> begin_to_apply	- nocondition -	Begin	Leave_Apply
<input type="checkbox"/> cancel_request	python:instance.requested != 'ok'	Refinement	End
<input type="checkbox"/> refinement_begin	python:instance.requested == 'ok'	Refinement	Begin

The fields marked A, F and G in the Activities-Properties-General tab correspond to the elements marked A, F and G in the JaWE2Openflow activities that are created in the activities screen.

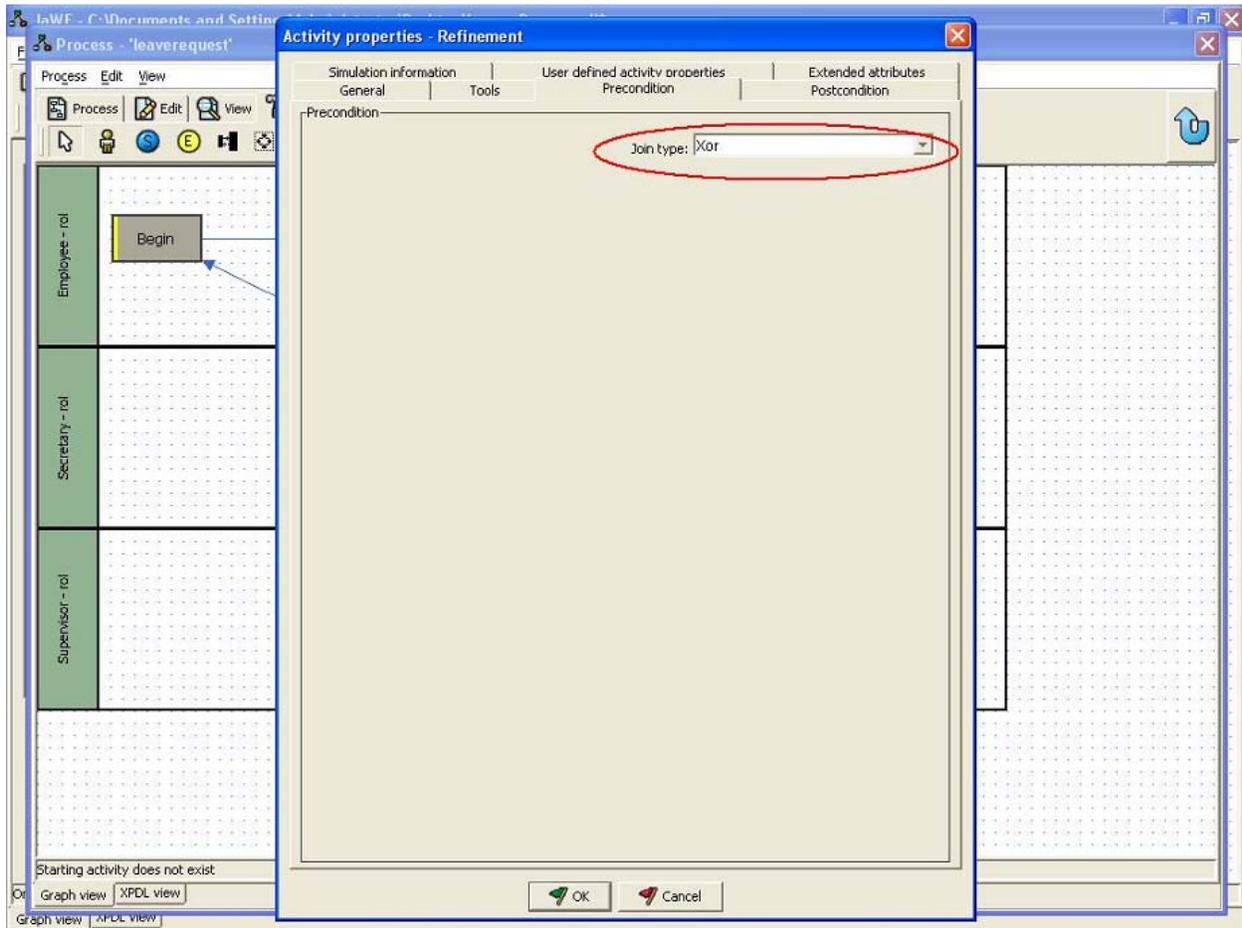


In the Activity-Properties-Tools tab, If the type is set to “Application,” it is considered as an “Application” as shown in the Activities list marked D in the JaWE2Openflow Activities snapshot and this convention is adopted by JaWE2Openflow product.

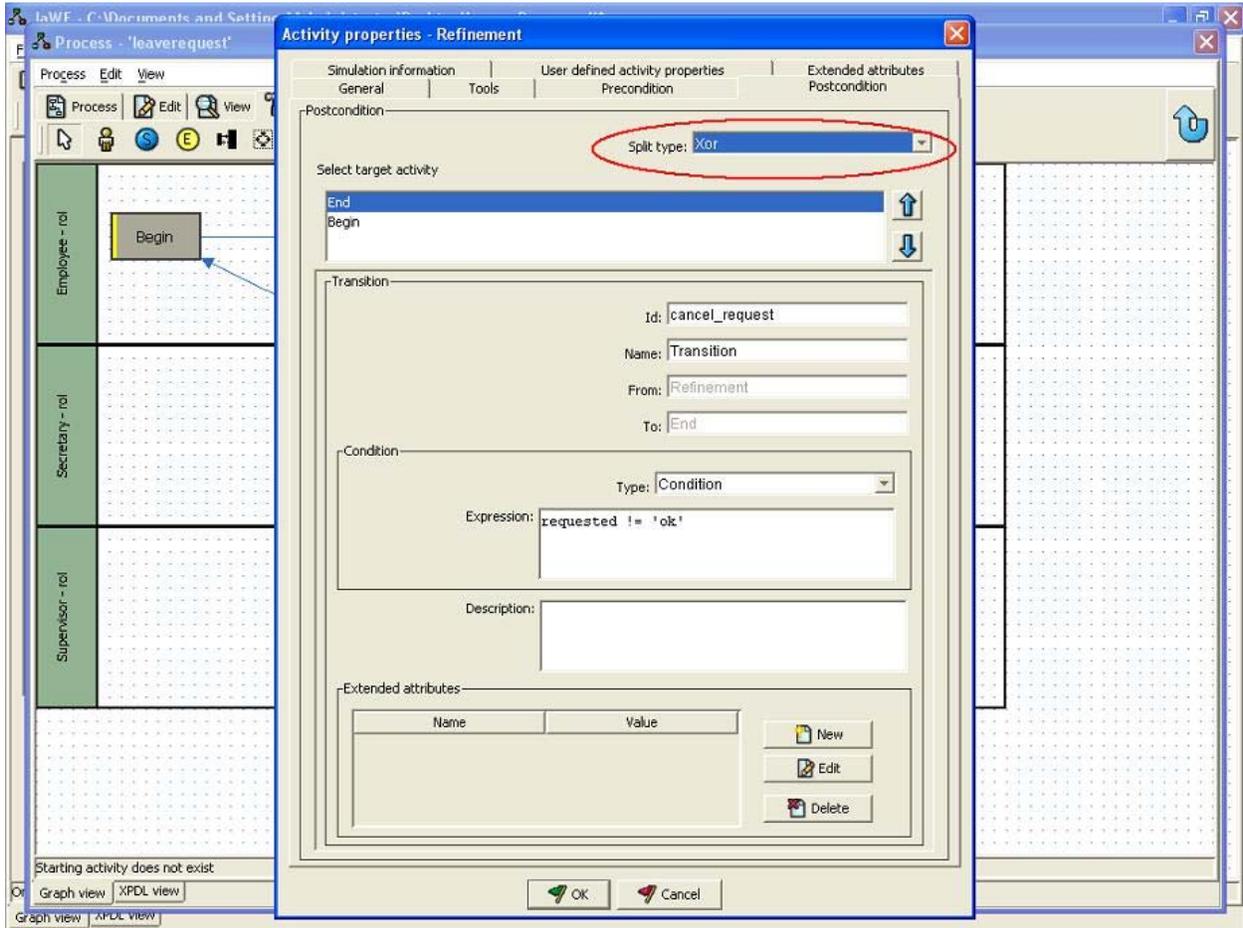
If the type is set to “Procedure” then it is considered a “Push Application” as shown in the Activities list marked E in the JaWE2Openflow Activities screen and this convention is adopted by the JaWE2Openflow product.



In the Activity-Properties-Precondition tab, the value of Join type is considered and is listed in the Activities list under the Join marked B in the JaWE2Openflow activities screen.



In the Activity-Properties-Post condition tab the value of Split type is considered and is listed in the Activities list under the Split heading marked C in the JaWE2Openflow activities screen.



6.6 Transitions

Arrows that appear between activities in the XPDL graph in the JaWE editor are considered to be transitions. Transitions are listed below the activities in the JaWE2Openflow screen.

The screenshot shows the Zope web interface in a Microsoft Internet Explorer browser window. The address bar shows `http://localhost:9000/manage`. The page is titled "Zope on http://localhost:9000 - Microsoft Internet Explorer" and is logged in as "john".

The main content area is divided into two sections: "Activities" and "Transitions".

Activities Table:

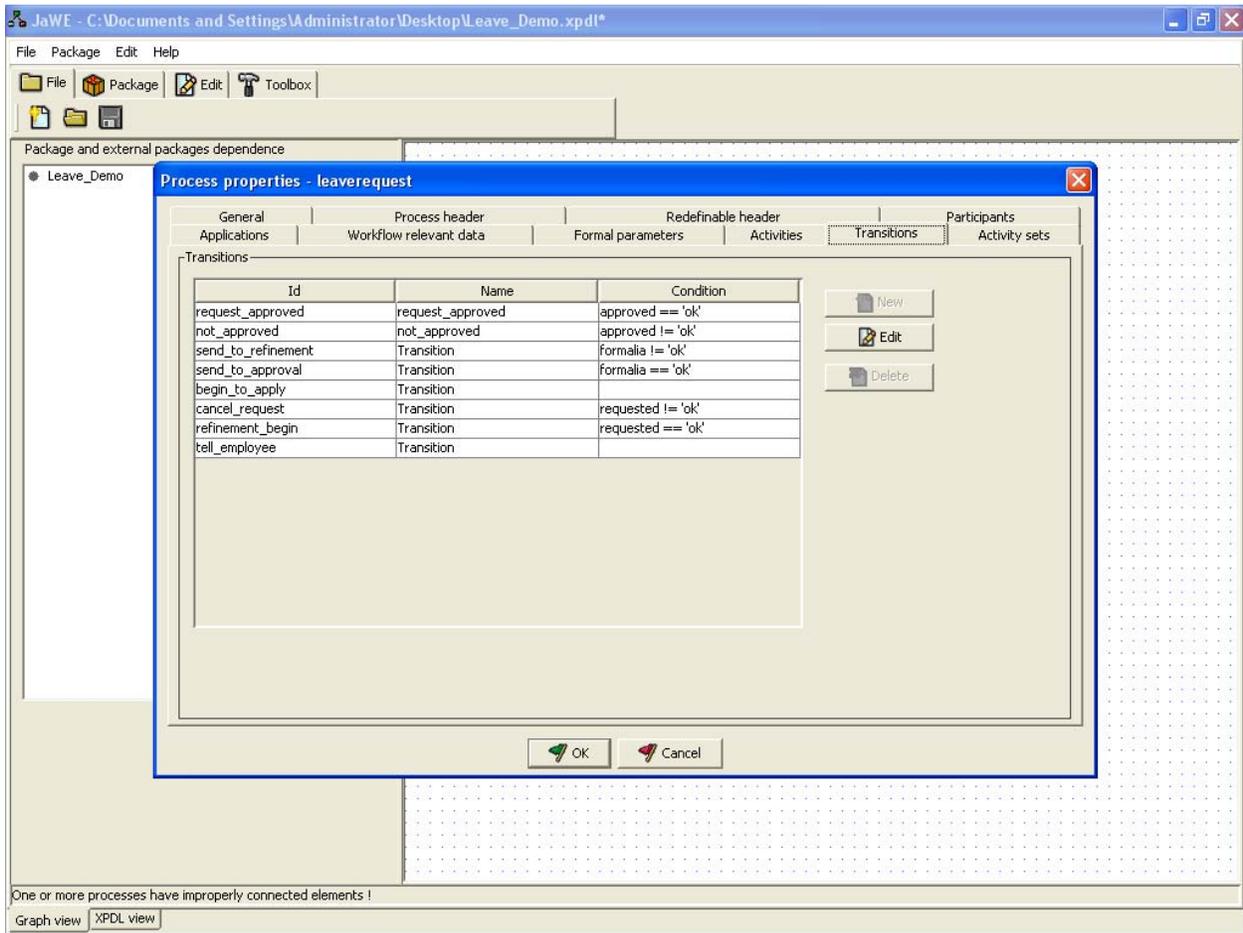
Activity	Kind	Join	Split	Application name	Push Application	Start mode	Finish mode	Subflow process
<input type="checkbox"/> Approval	standard	and	xor	leave_approvalfom	route_to_supervisor	Manual	Automatic	
<input type="checkbox"/> UpdateHR	standard	and	and	leave_hrfom	route_to_secretary	Manual	Automatic	
<input type="checkbox"/> Refinement	standard	xor	xor	leave_refine	route_to_customer	Manual	Automatic	
<input type="checkbox"/> Leave_Apply	standard	and	xor	leave_checkstatus	route_to_secretary	Manual	Automatic	
<input type="checkbox"/> Begin	standard	and	and			Manual	Automatic	
<input type="checkbox"/> End	standard	xor	and	leave_finainfo	route_to_customer	Manual	Manual	

Transitions Table:

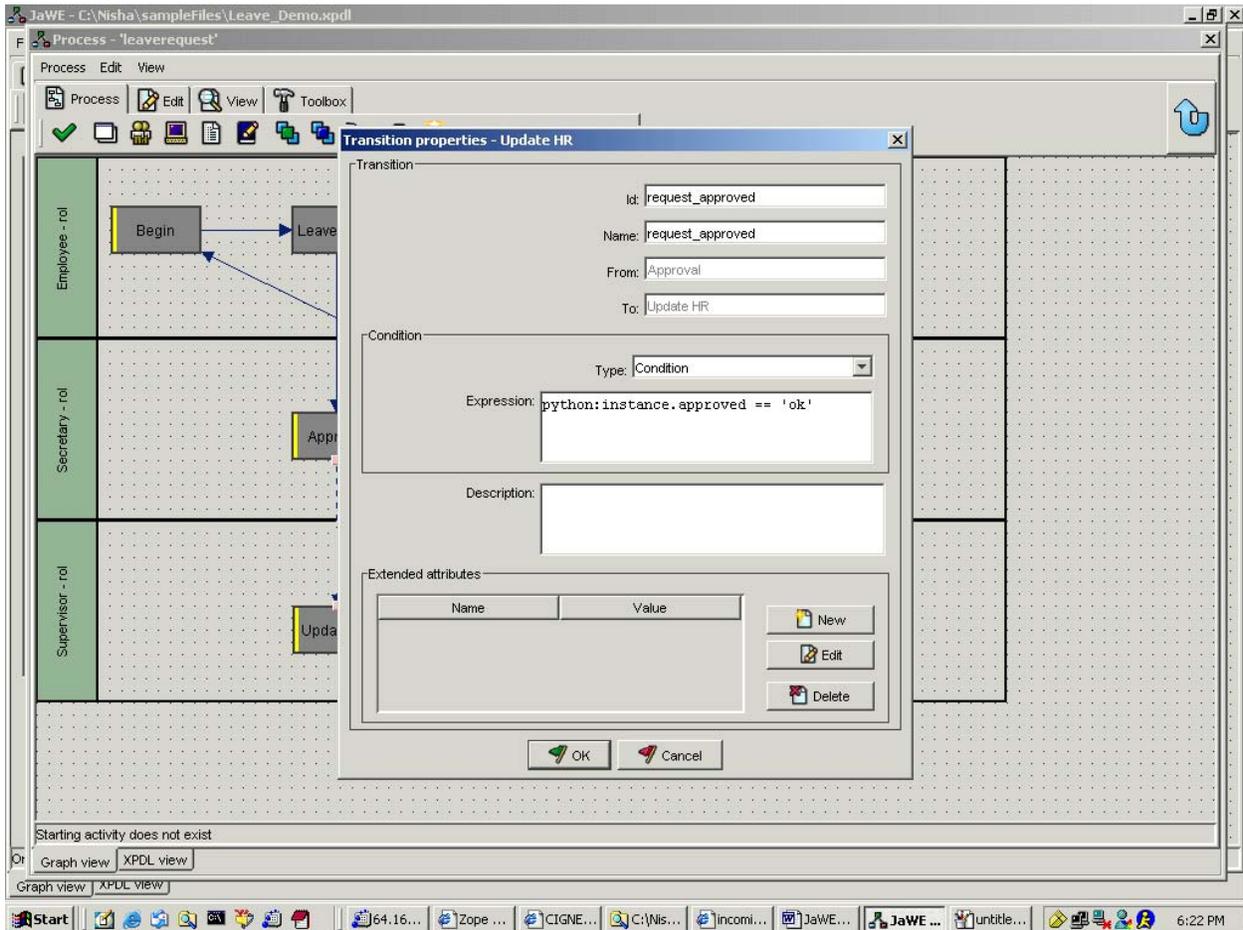
Transition	Condition	From	To
<input type="checkbox"/> request_approved	<code>python:instance.approved == 'ok'</code>	Approval	UpdateHR
<input type="checkbox"/> not_approved	<code>python:instance.approved != 'ok'</code>	Approval	Refinement
<input type="checkbox"/> send_to_refinement	<code>python:instance.formalia != 'ok'</code>	Leave_Apply	Refinement
<input type="checkbox"/> send_to_approval	<code>python:instance.formalia == 'ok'</code>	Leave_Apply	Approval
<input type="checkbox"/> begin_to_apply	- nocondition -	Begin	Leave_Apply
<input type="checkbox"/> cancel_request	<code>python:instance.requested != 'ok'</code>	Refinement	End
<input type="checkbox"/> refinement_begin	<code>python:instance.requested == 'ok'</code>	Refinement	Begin
<input type="checkbox"/> tell_employee	- nocondition -	UpdateHR	End

The "Transitions" table has red circles around the "Transition", "Condition", "From", and "To" columns of the first row.

The list of transitions at the Process-Properties level shown below was created in JaWE.



In the Transition Properties screen, all of the fields marked are considered when creating JaWE2Openflow object.



In JaWE2Openflow, the transitions display below the activities on the Map tab.

The screenshot shows the ZOPE web interface in Microsoft Internet Explorer. The browser address bar shows 'http://localhost:9000/manage'. The ZOPE interface is logged in as 'john'. The 'Map' tab is selected, showing the process path: '/xpd2openflow/site/AAA111_Folder/AAA111/leaverequest'. There are 'Add Activity' and 'Add Transition' buttons. Below are two tables: 'Activities' and 'Transitions'.

Activities

Activity	Kind	Join	Split	Application name	Push Application	Start mode	Finish mode	Subflow process
<input type="checkbox"/> Approval	standard	and	xor	leave_approvalform	route_to_supervisor	Manual	Automatic	
<input type="checkbox"/> UpdateHR	standard	and	and	leave_hrform	route_to_secretary	Manual	Automatic	
<input type="checkbox"/> Refinement	standard	xor	xor	leave_refine	route_to_customer	Manual	Automatic	
<input type="checkbox"/> Leave_Apply	standard	and	xor	leave_checkstatus	route_to_secretary	Manual	Automatic	
<input type="checkbox"/> Begin	standard	and	and			Manual	Automatic	
<input type="checkbox"/> End	standard	xor	and	leave_finainfo	route_to_customer	Manual	Manual	

Transitions

Transition	Condition	From	To
<input type="checkbox"/> request_approved	python:instance.approved == 'ok'	Approval	UpdateHR
<input type="checkbox"/> not_approved	python:instance.approved != 'ok'	Approval	Refinement
<input type="checkbox"/> send_to_refinement	python:instance.formalia != 'ok'	Leave_Apply	Refinement
<input type="checkbox"/> send_to_approval	python:instance.formalia == 'ok'	Leave_Apply	Approval
<input type="checkbox"/> begin_to_apply	- nocondition -	Begin	Leave_Apply
<input type="checkbox"/> cancel_request	python:instance.requested != 'ok'	Refinement	End
<input type="checkbox"/> refinement_basis	python:instance.requested == 'ok'	Refinement	Begin

7 Sample Conversion Process

7.1 Scenario

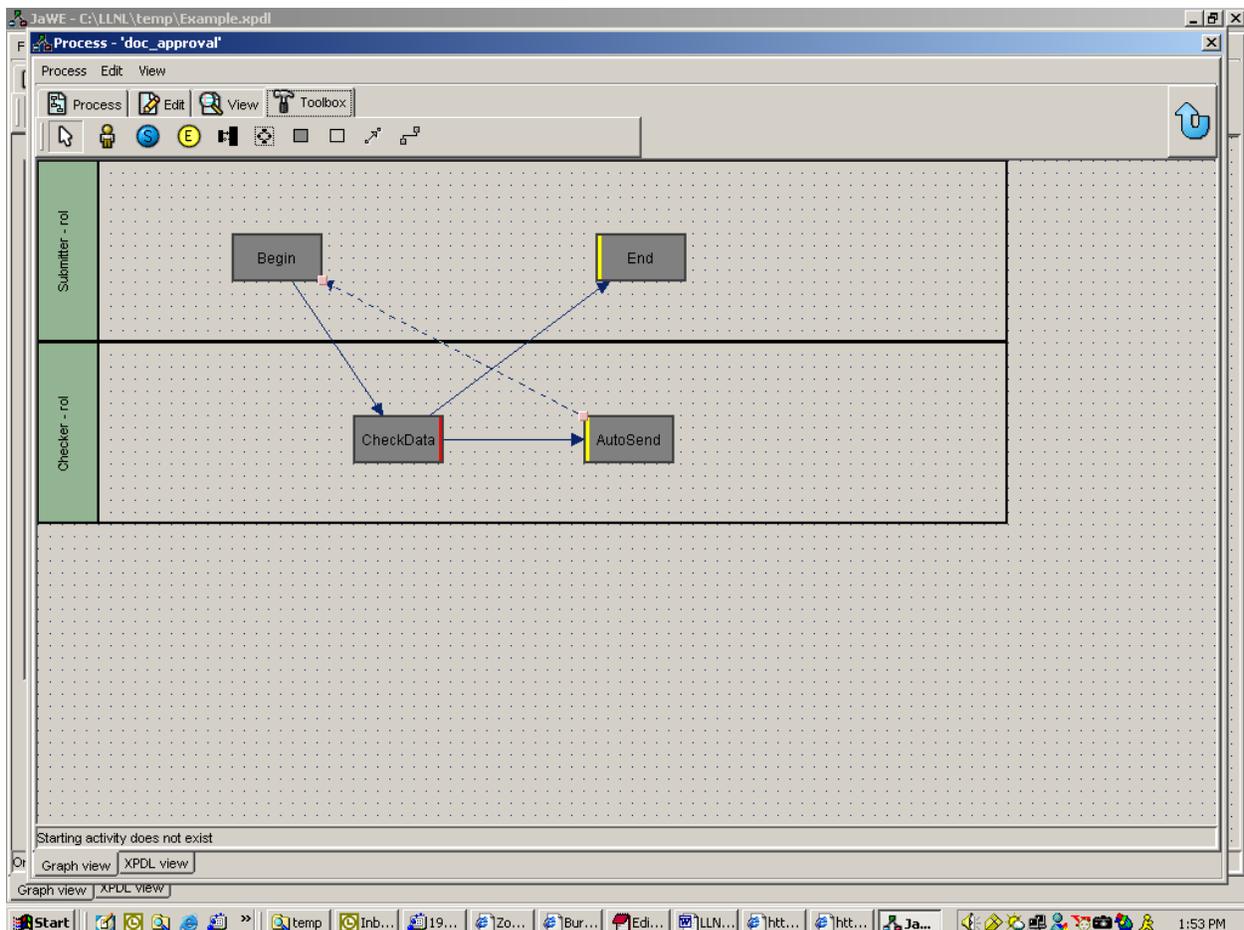
This section describes a simple OpenFlow workflow application.

The scenario is a simple workflow process where a submitter submits a document and a checker checks the document and either approves it or rejects it. Rejected document will be sent back to the submitter. The submitter can modify the document and upload it again for review. If the document is approved, the “doc_approved” property will be set to “Yes” for the workflow instance.

Sample files are included with the JaWE2Openflow product and they are in the product’s “sampleFiles” directory. It has two files:

1. Workflow_folder_customized.zexp is an export file for the above-mentioned application.
2. Workflow.xpdl is a file created in JaWE and needs some customization.

7.2 JaWE Screen for Workflow.xpdl



7.3 Workflow.XPDL (XML Dump)

```
<?xml version="1.0" encoding="UTF-8"?>
<Package Id="Example" xmlns="http://www.wfmc.org/2002/XPDL1.0"
xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0 http://wfmc.org/standards/docs/TC-
1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>1.0</XPDLVersion>
    <Vendor>Together</Vendor>
    <Created>2004-06-09 16:06:59</Created>
  </PackageHeader>
  <RedefinableHeader PublicationStatus="UNDER_TEST"/>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <WorkflowProcesses>
    <WorkflowProcess AccessLevel="PUBLIC" Id="doc_approval" Name="doc_approval">
      <ProcessHeader DurationUnit="D">
        <Created>2004-06-09 16:08:14</Created>
      </ProcessHeader>
      <RedefinableHeader PublicationStatus="UNDER_TEST"/>
      <Participants>
        <Participant Id="Checker">
          <ParticipantType Type="ROLE"/>
        </Participant>
        <Participant Id="Submitter">
          <ParticipantType Type="ROLE"/>
        </Participant>
      </Participants>
      <Applications>
        <Application Id="check_dataForm" Name="check_dataForm">
          <Description>This is check data form</Description>
        </Application>
        <Application Id="data_rejected" Name="data_rejected"/>
        <Application Id="data_approved" Name="data_approved"/>
        <Application Id="doc_edit" Name="doc_edit"/>
      </Applications>
      <Activities>
        <Activity Id="Begin" Name="Begin">
          <Implementation>
            <Tool Id="doc_edit" Type="APPLICATION"/>
          </Implementation>
          <Performer>Submitter</Performer>
          <StartMode>
            <Manual/>
          </StartMode>
          <FinishMode>
            <Automatic/>
          </FinishMode>
          <ExtendedAttributes>
            <ExtendedAttribute Name="ParticipantID" Value="Submitter"/>
            <ExtendedAttribute Name="XOffset" Value="160"/>
            <ExtendedAttribute Name="YOffset" Value="60"/>
          </ExtendedAttributes>
        </Activity>
        <Activity Id="CheckData" Name="CheckData">
```

```

<Implementation>
  <Tool Id="check_dataForm" Type="APPLICATION"/>
</Implementation>
<Performer>Checker</Performer>
<StartMode>
  <Manual/>
</StartMode>
<FinishMode>
  <Automatic/>
</FinishMode>
<TransitionRestrictions>
  <TransitionRestriction>
    <Split Type="XOR">
      <TransitionRefs>
        <TransitionRef Id="CheckData_End"/>
        <TransitionRef Id="CheckData_AutoSend"/>
      </TransitionRefs>
    </Split>
  </TransitionRestriction>
</TransitionRestrictions>
<ExtendedAttributes>
  <ExtendedAttribute Name="ParticipantID" Value="Checker"/>
  <ExtendedAttribute Name="XOffset" Value="260"/>
  <ExtendedAttribute Name="YOffset" Value="60"/>
</ExtendedAttributes>
</Activity>
<Activity Id="End" Name="End">
  <Implementation>
    <Tool Id="data_approved" Type="APPLICATION"/>
  </Implementation>
  <Performer>Submitter</Performer>
  <StartMode>
    <Automatic/>
  </StartMode>
  <FinishMode>
    <Automatic/>
  </FinishMode>
  <ExtendedAttributes>
    <ExtendedAttribute Name="ParticipantID" Value="Submitter"/>
    <ExtendedAttribute Name="XOffset" Value="460"/>
    <ExtendedAttribute Name="YOffset" Value="60"/>
  </ExtendedAttributes>
</Activity>
<Activity Id="AutoSend" Name="AutoSend">
  <Implementation>
    <Tool Id="data_rejected" Type="APPLICATION"/>
  </Implementation>
  <Performer>Checker</Performer>
  <StartMode>
    <Automatic/>
  </StartMode>
  <FinishMode>
    <Automatic/>
  </FinishMode>
  <ExtendedAttributes>
    <ExtendedAttribute Name="ParticipantID" Value="Checker"/>
  </ExtendedAttributes>

```

```

        <ExtendedAttribute Name="XOffset" Value="450"/>
        <ExtendedAttribute Name="YOffset" Value="60"/>
    </ExtendedAttributes>
</Activity>
</Activities>
<Transitions>
    <Transition From="Begin" Id="Begin_CheckData" Name="Begin_CheckData" To="CheckData">
        <ExtendedAttributes>
            <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
        </ExtendedAttributes>
    </Transition>
    <Transition From="CheckData" Id="CheckData_AutoSend" Name="CheckData_AutoSend"
To="AutoSend">
        <Condition Type="CONDITION">python:not(instance.getProperty('approved', ''))</Condition>
        <ExtendedAttributes>
            <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
        </ExtendedAttributes>
    </Transition>
    <Transition From="CheckData" Id="CheckData_End" Name="CheckData_End" To="End">
        <Condition Type="CONDITION">python:instance.getProperty('approved', '')</Condition>
        <ExtendedAttributes>
            <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
        </ExtendedAttributes>
    </Transition>
    <Transition From="AutoSend" Id="AutoSend_Begin" Name="AutoSend_Begin" To="Begin">
        <ExtendedAttributes>
            <ExtendedAttribute Name="RoutingType" Value="NOROUTING"/>
        </ExtendedAttributes>
    </Transition>
</Transitions>
<ExtendedAttributes>
    <ExtendedAttribute Name="ParticipantVisualOrder" Value="Submitter;Checker;"/>
</ExtendedAttributes>
</WorkflowProcess>
</WorkflowProcesses>
<ExtendedAttributes>
    <ExtendedAttribute Name="MadeBy" Value="JaWE"/>
    <ExtendedAttribute Name="Version" Value="1.2"/>
</ExtendedAttributes>
</Package>

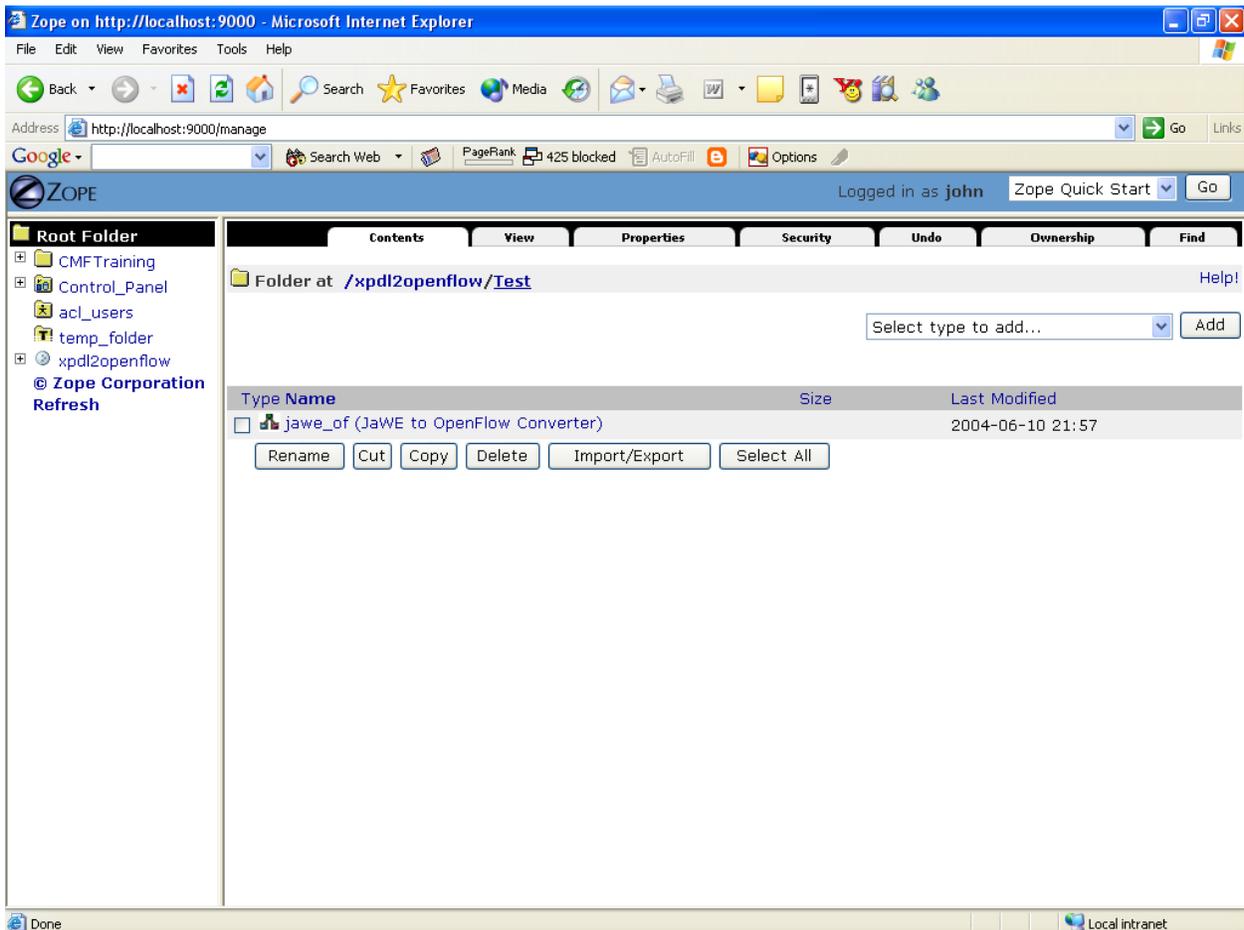
```

7.4 Upload Workflow.xpdl file

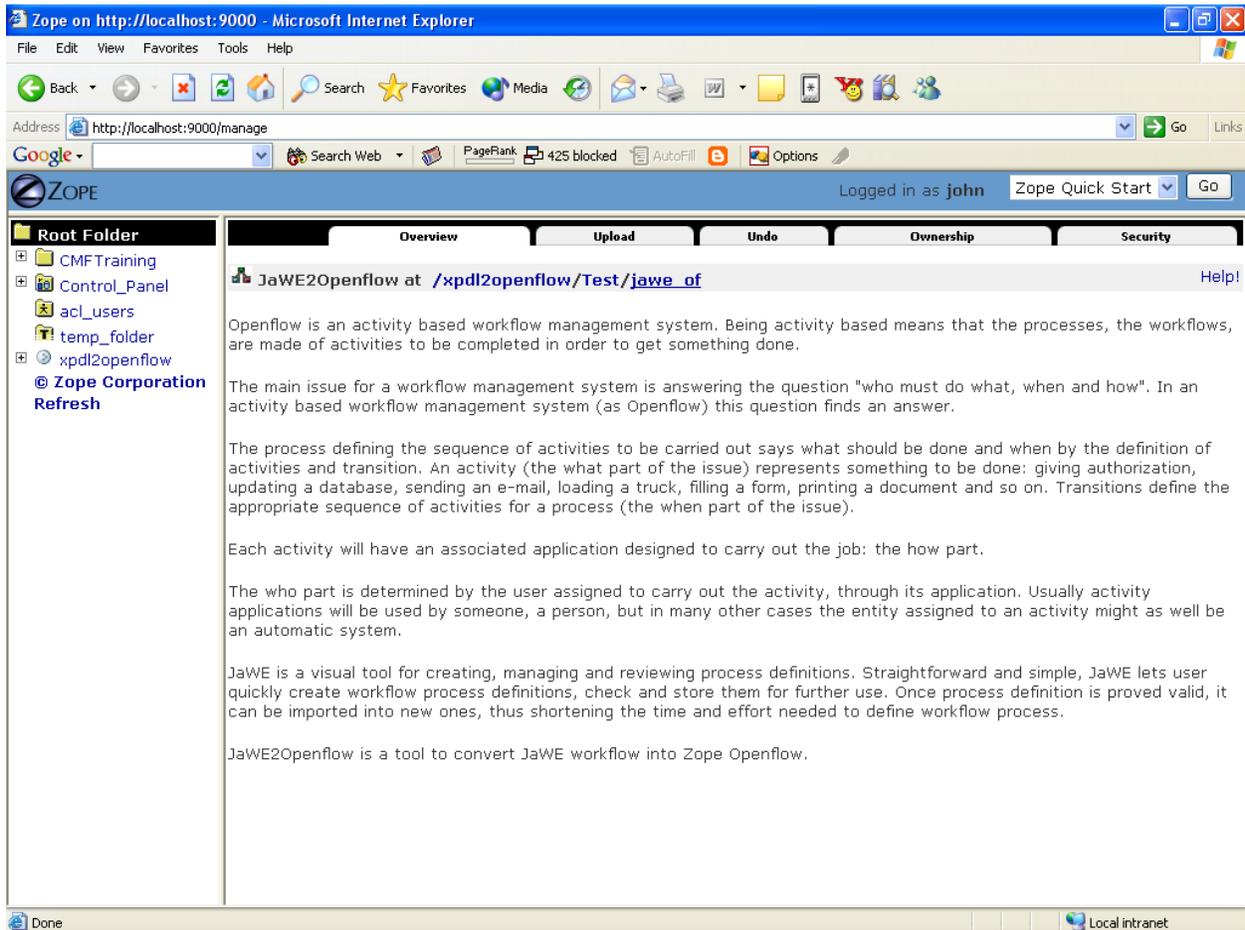
Follow the steps below to upload the workflow.xpdl file.

To upload the workflow.xpdl file:

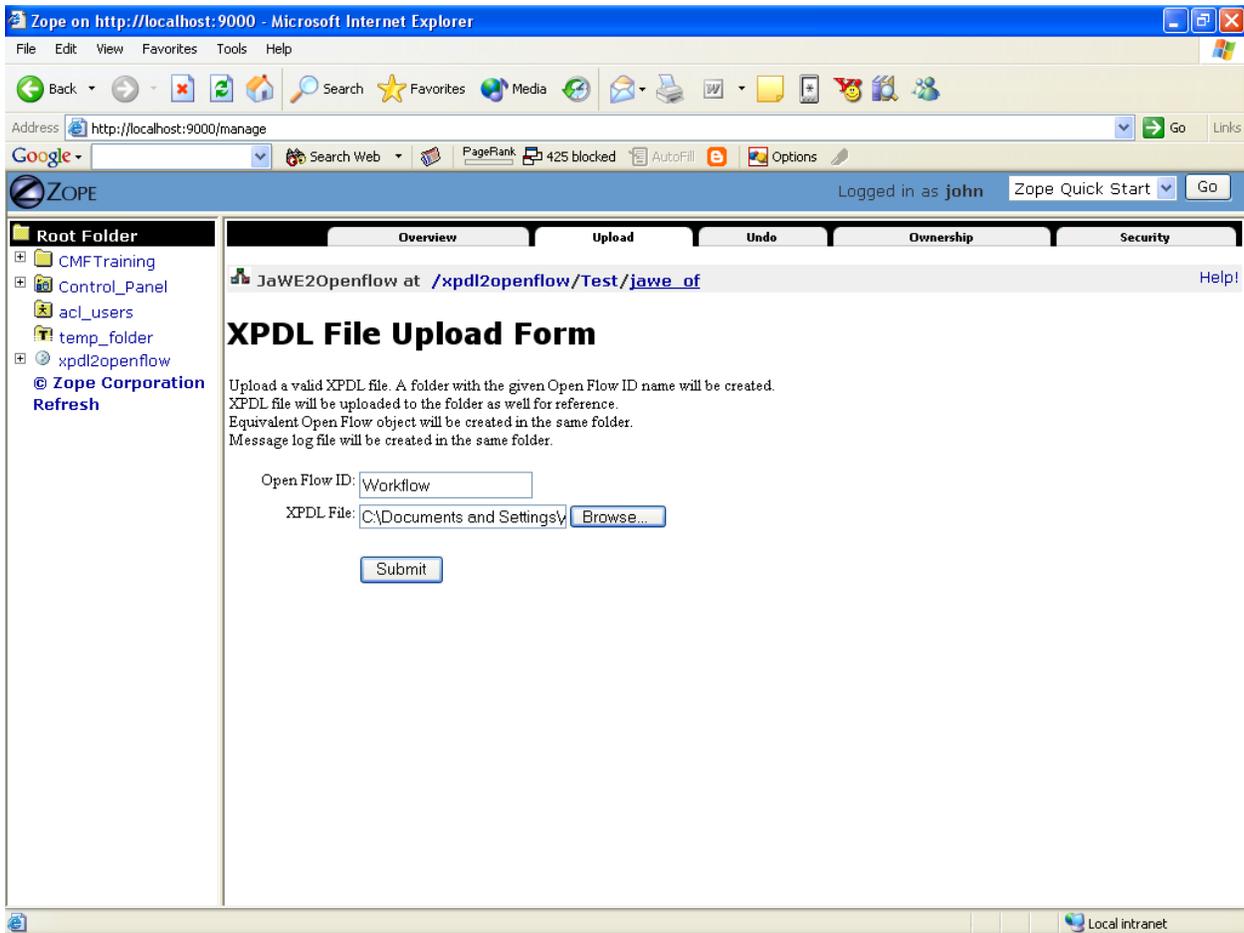
1. Copy/Save the workflow.xpdl file from the JaWE2Openflow sampleFiles directory to a local machine.
2. Install the JaWE2Openflow product (if not already installed).
3. Add an instance of the JaWE2Openflow by selecting it from the select type to add list of Zope objects and click on the Add button. An instance of the JaWE2Openflow is created with the default name “jawe_of” that can be given at the time of adding the object.
4. If an instance of the JaWE2Openflow product exists, use the upload tab in the instance to upload an XPDL file.



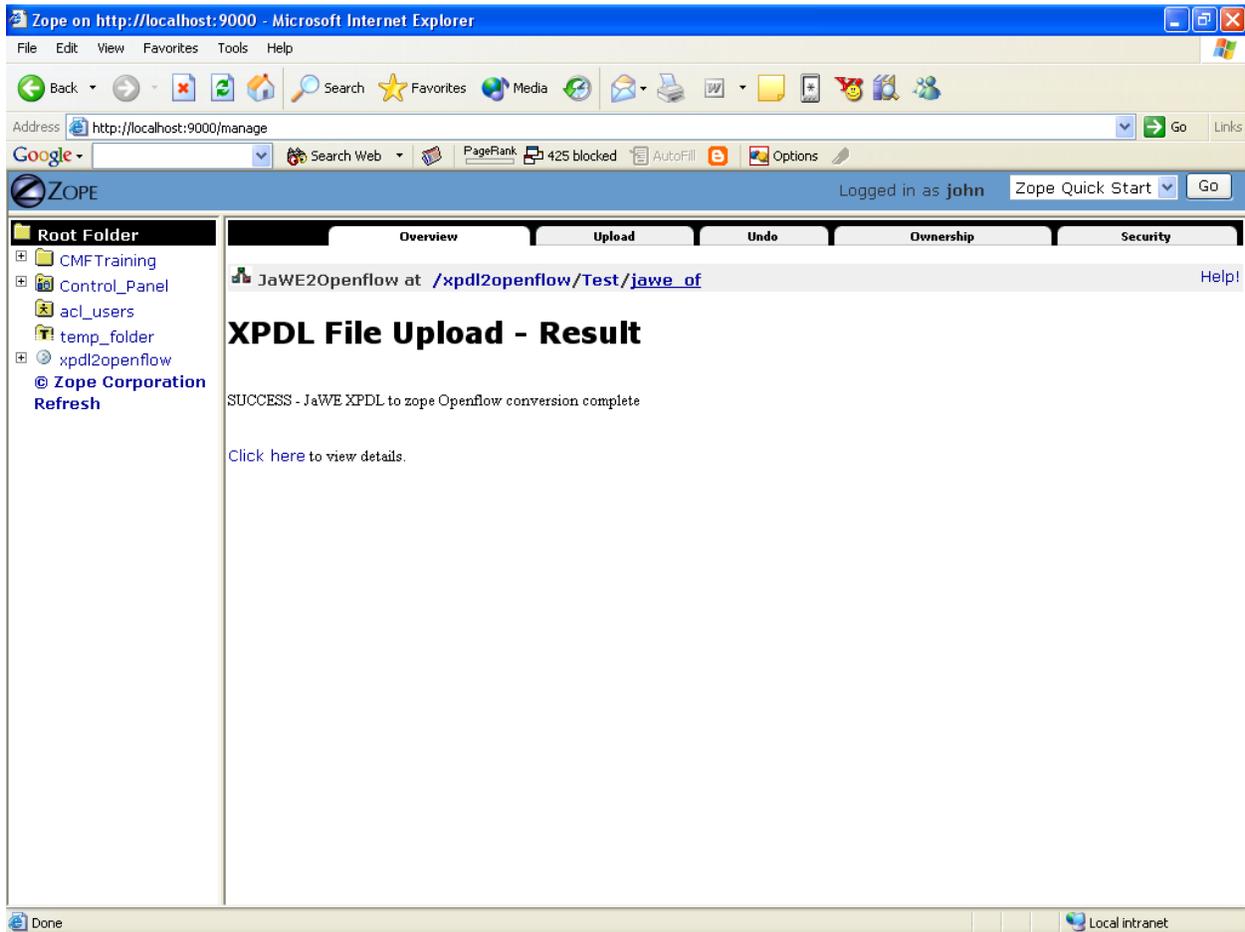
5. Click on the JaWE2Openflow instance object, a ZMI screen displays with the description of JaWE2Openflow.



- Click on the Upload tab. A ZMI screen with two data fields “Open Flow ID” and “XPDL file” appears. Enter “Workflow” as the ID for OpenFlow. Click the Browse button, select the workflow.xpdl file from the local machine, and click the Submit button.



7. A results screen with a message appears with a link to the message log file. Click on the link to view the details.

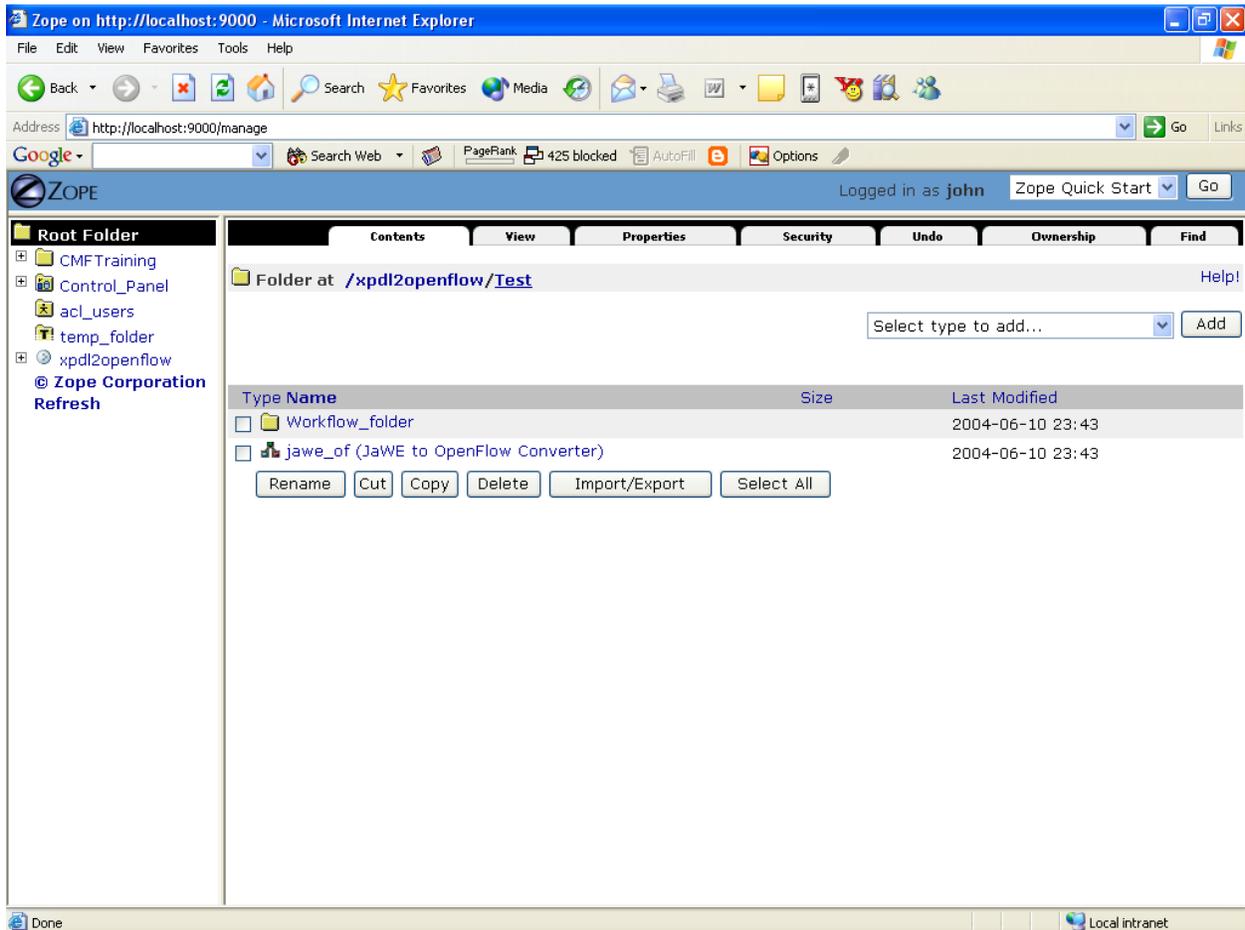


8. The sequence of steps while converting the XPDL file to OpenFlow using JaWE2Openflow displays.

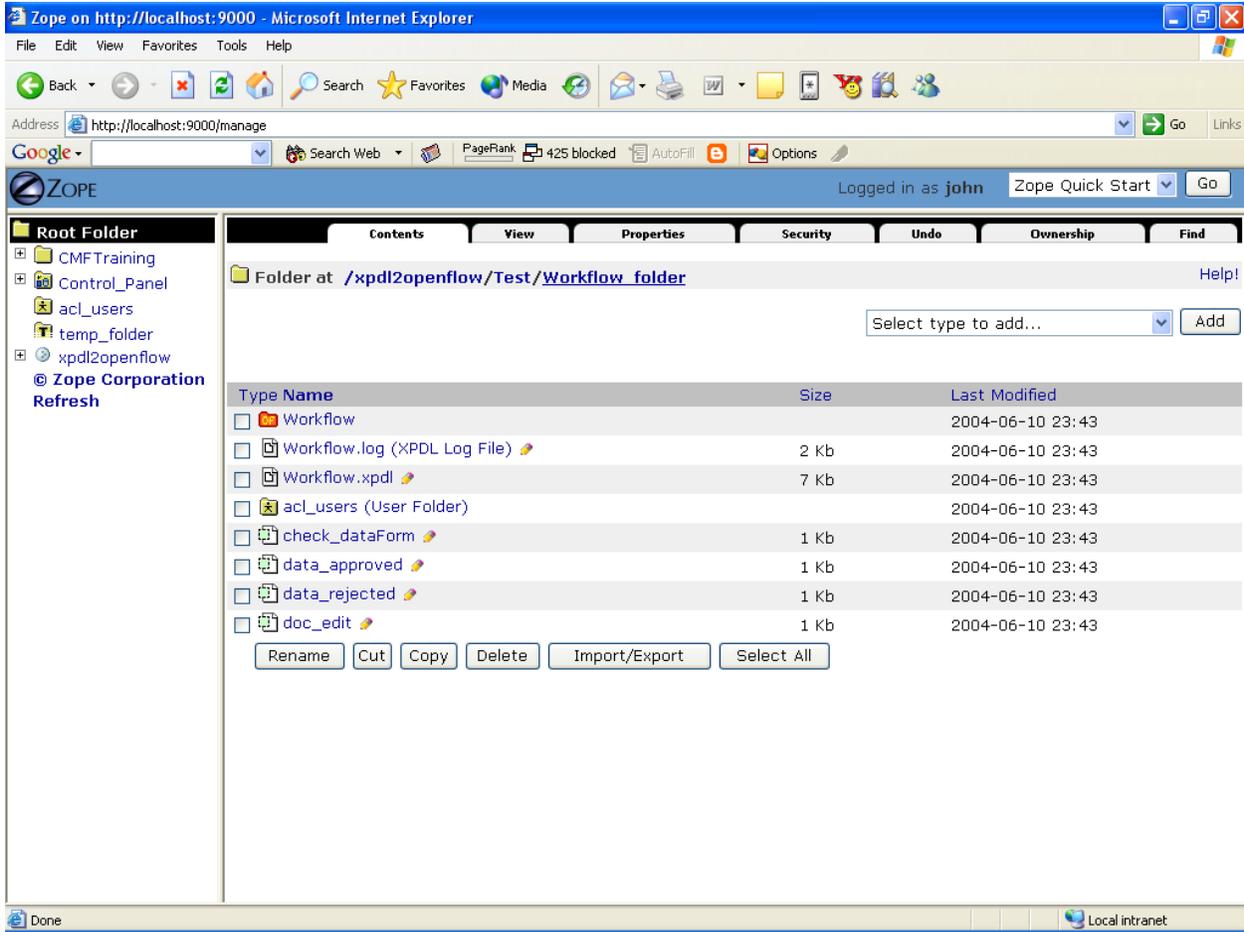


```
Workflow[1] - Notepad
File Edit Format View Help
Uploaded File: C:\Documents and Settings\Administrator\Desktop\workflow.xpdl
By: john
Timestamp: 2004/06/10
Folder: workflow_folder
Log File: workflow.log
*****
INFO: Total Applications are - 4
*****
Adding Applications...
*****
check_dataForm application is added in zope openflow.
data_rejected application is added in zope openflow.
data_approved application is added in zope openflow.
doc_edit application is added in zope openflow.
INFO: Total Package Participants are - 2
*****
Adding Package level Participants...
*****
Checker role is added.
Checker user is added in acl_users folder.
Submitter role is added.
Submitter user is added in acl_users folder.
INFO: Total Processes are - 1
Creating Process doc_approval
*****
INFO: Total Activities are - 4
*****
Adding Activities...
*****
Begin activity is added to zope openflow.
CheckData activity is added to zope openflow.
End activity is added to zope openflow.
AutoSend activity is added to zope openflow.
INFO: Total Transitions are - 4
*****
Adding Transitions...
*****
Begin_CheckData transition is added to zope openflow.
CheckData_AutoSend transition is added to zope openflow.
CheckData_End transition is added to zope openflow.
AutoSend_Begin transition is added to zope openflow.
doc_approval Process Created
*****
SUCCESS - JaWE XPDL to zope openflow conversion complete
```

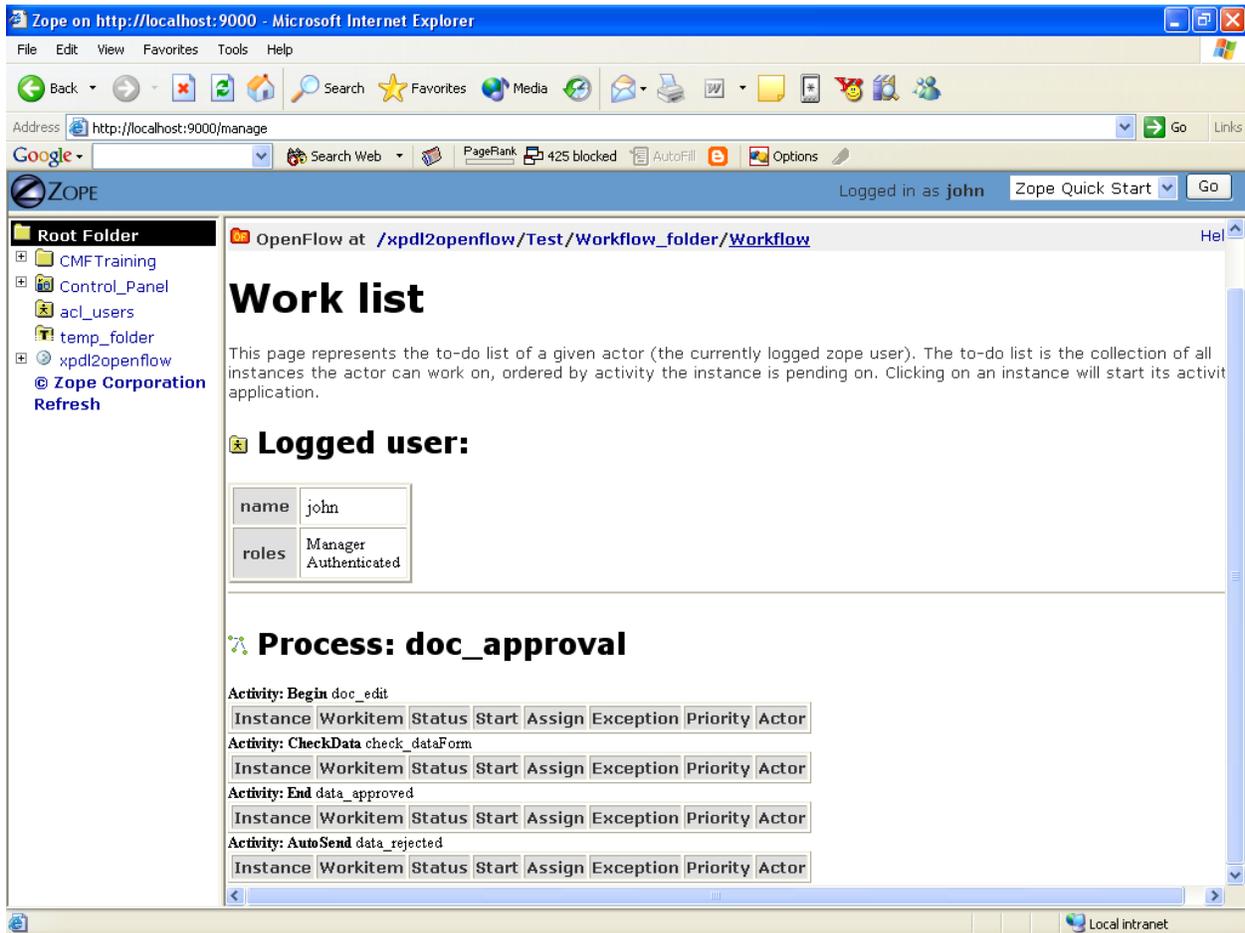
9. A folder is created at the JaWE2Openflow instance folder. The name of the folder will match the name specified as the OpenFlow ID with “_folder” added to the end.



- Click on the folder created. An OpenFlow object with the given name is created. The `acl_users` folder is created with user names matching the participants. Roles are also created in the Security tab that match the participants. A log file is created where all XPDL file to OpenFlow conversion steps are logged along with the exceptions raised and the errors that occurred during the conversion. The uploaded XPDL file is also added.



11. Click on the OpenFlow object in the Worklist tab to view the processes and activities in the process.

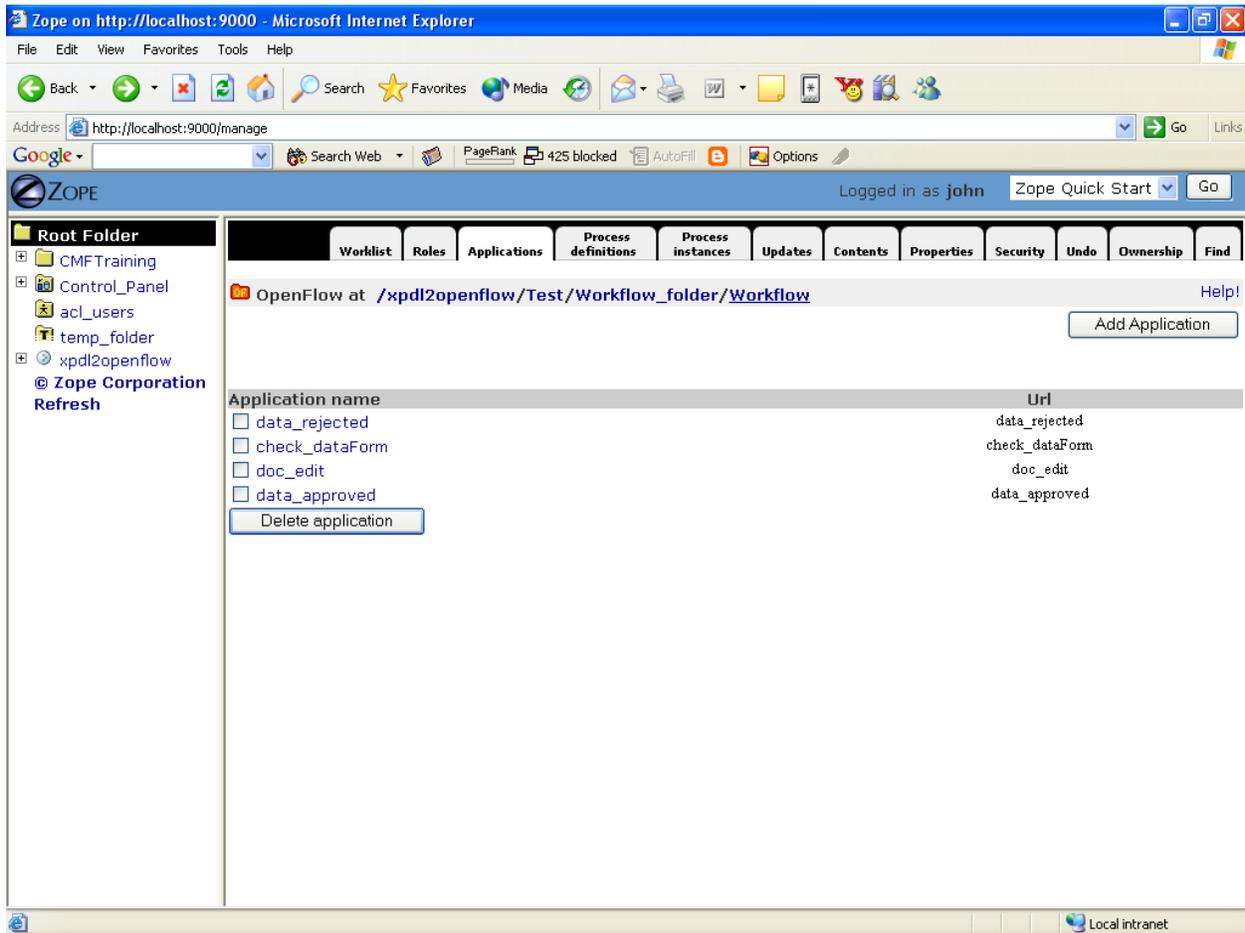


12. Click on the Roles tab to view the roles with the participants in the XPDL file.

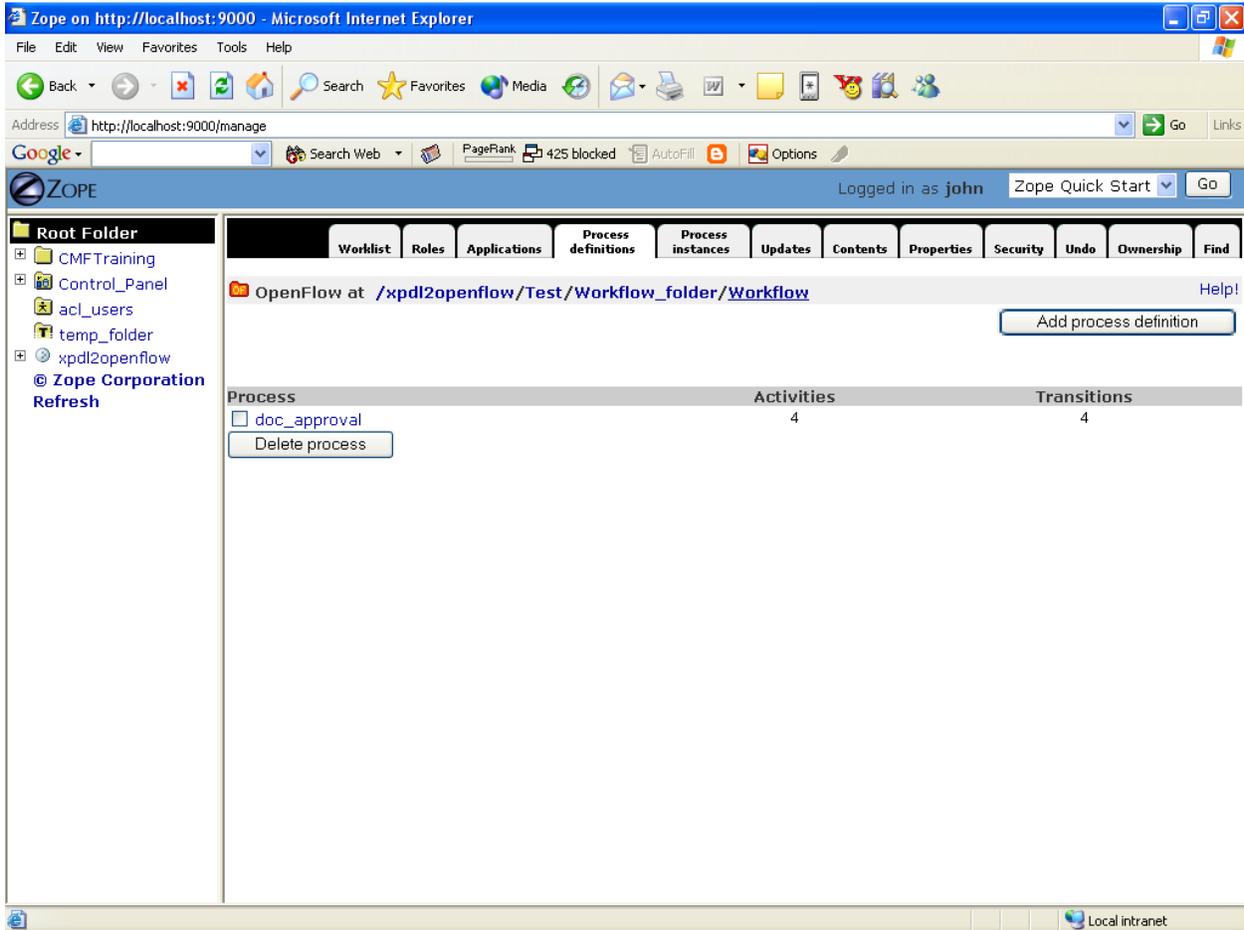
The screenshot shows the ZOPE web interface in Microsoft Internet Explorer. The browser address bar shows `http://localhost:9000/manage`. The ZOPE header indicates the user is logged in as 'john'. The left sidebar shows a tree view under 'Root Folder' with items like 'CMF Training', 'Control_Panel', 'acl_users', 'temp_folder', and 'xpdlopenflow'. The main content area has a navigation menu with tabs: 'Worklist', 'Roles', 'Applications', 'Process definitions', 'Process instances', 'Updates', 'Contents', 'Properties', 'Security', 'Undo', 'Ownership', and 'Find'. The 'Roles' tab is active, displaying 'OpenFlow at /xpdlopenflow/Test/Workflow_folder/Workflow' and 'Manage Roles'. Below this is a table titled 'Roles' with three columns: 'Role', 'Pushable activities', and 'Pullable activities'. The table lists various roles, all with 'not assigned' in the activity columns.

Role	Pushable activities	Pullable activities
Anonymous	not assigned	not assigned
Authenticated	not assigned	not assigned
Checker	not assigned	not assigned
Manager	not assigned	not assigned
Member	not assigned	not assigned
Owner	not assigned	not assigned
Reviewer	not assigned	not assigned
Submitter	not assigned	not assigned

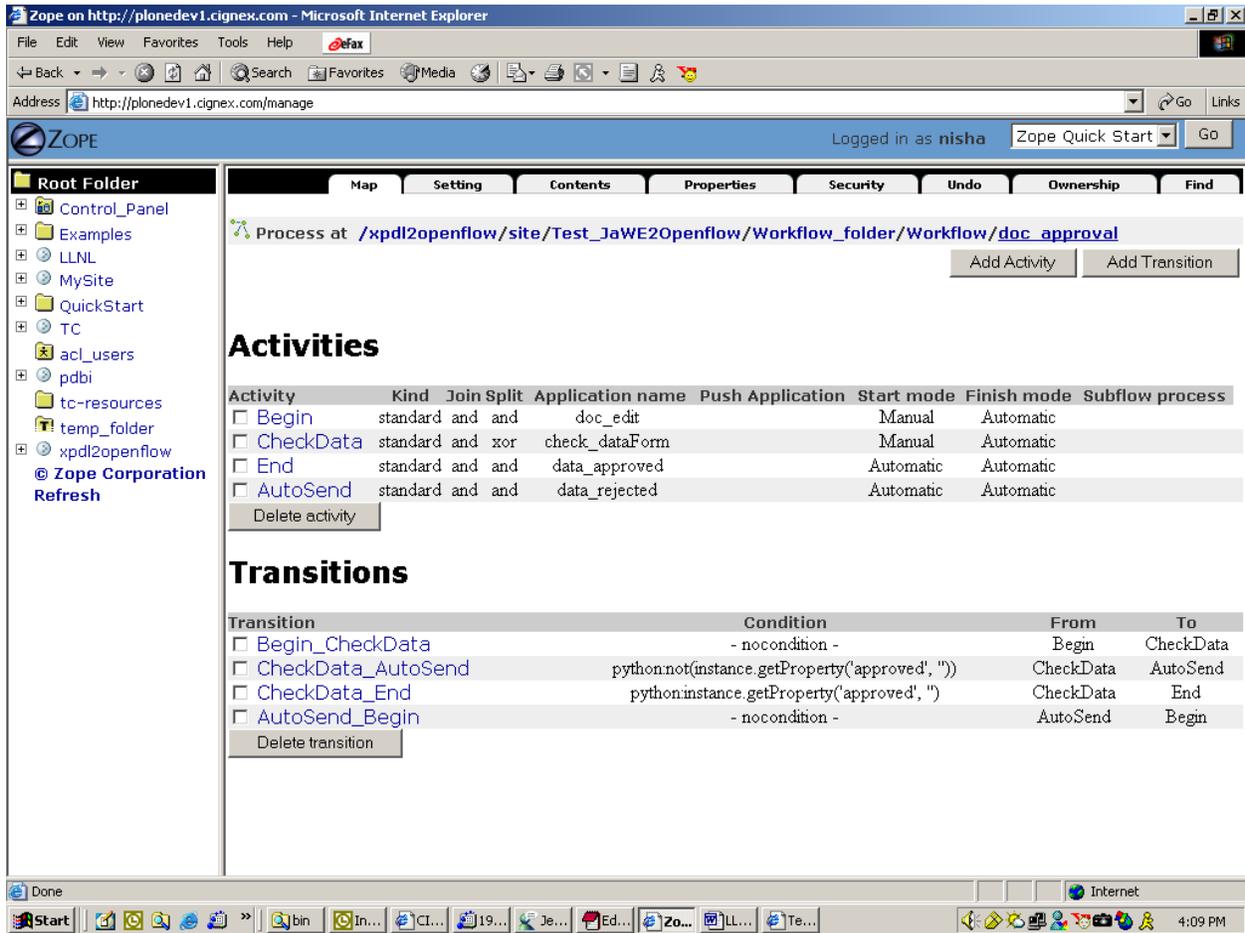
13. Click on the Applications tab where all of the applications are listed.



14. In the Process Definitions tab, all of the processes are listed with the number of activities and transitions in each process.



15. Click on a process to see a list of its activities and transitions.



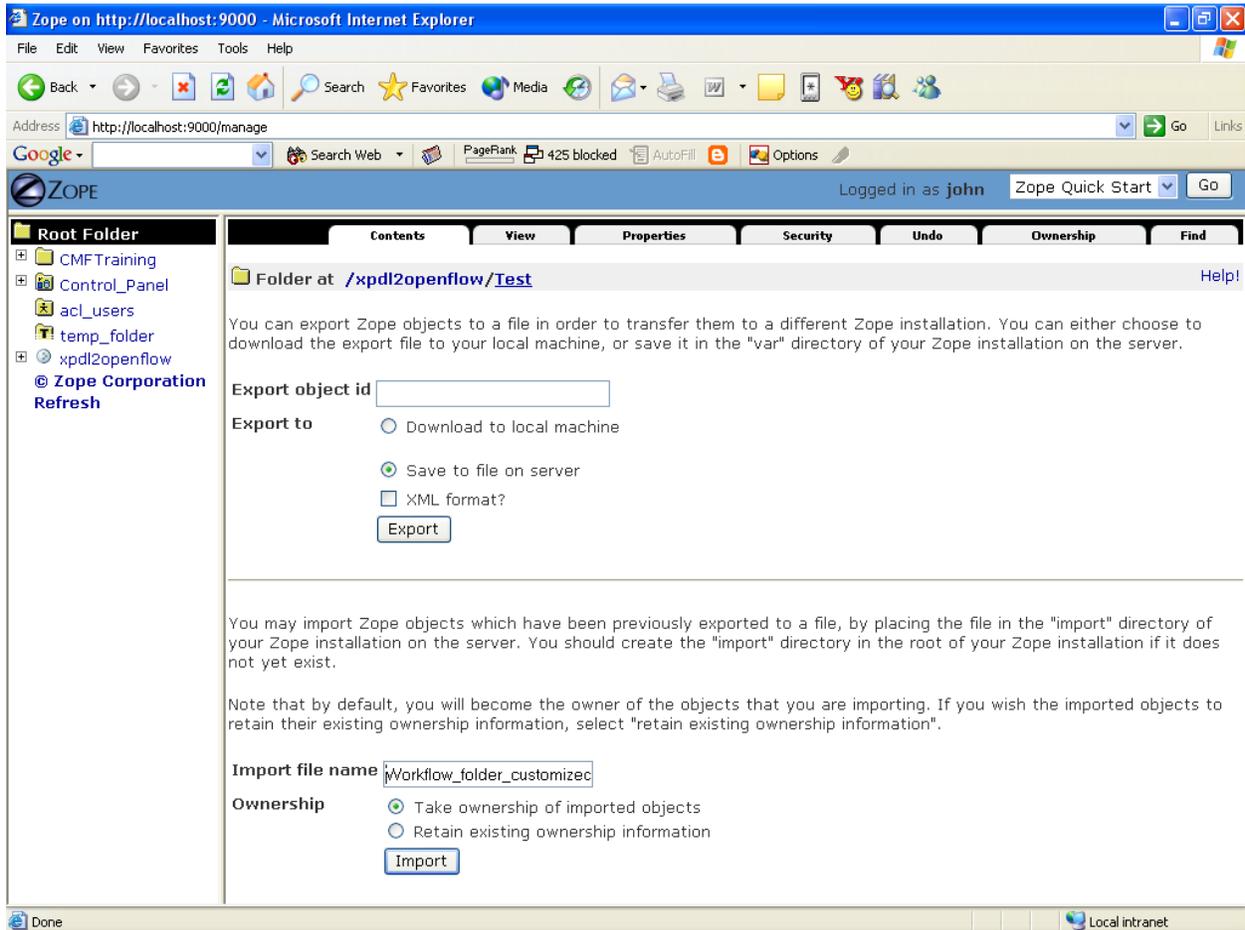
16. This application is ready to be customized as described in the next section (Section 7.5).

7.5 Customization

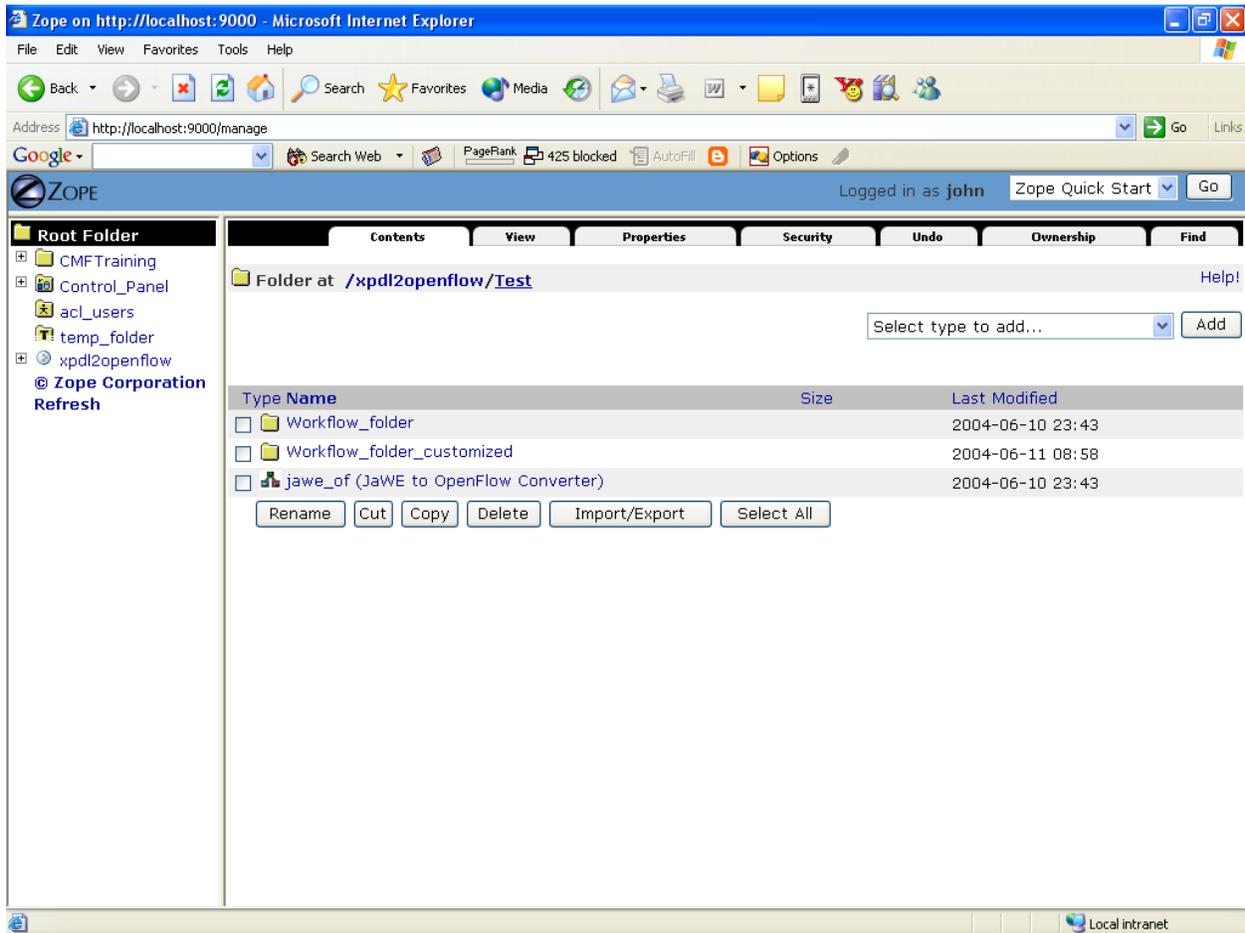
The sample application considered in the previous section is customized and supplied as .zexp file in the JaWE2Openflow sampleFiles folder named “Workflow_folder_customized.zexp”.

To customize the application:

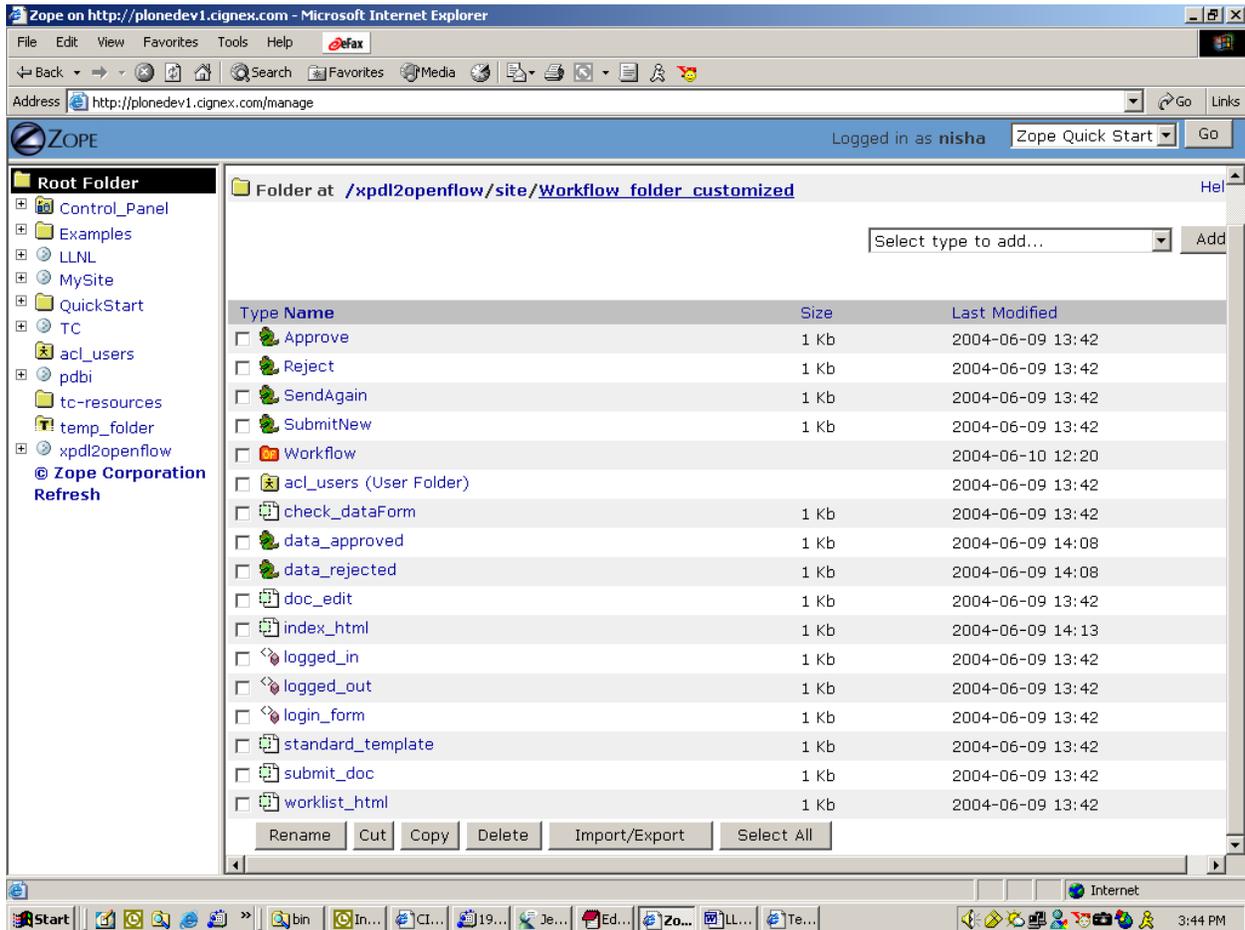
1. Copy the file into the <zope_instance>/import folder. Go to ZMI and proceed to the folder where JaWE2Openflow instance is present.
2. Click on the Import/Export button and type “Workflow_folder_customized.zexp” in the Import file name field. Click on the Import button.



3. The folder 'Workflow_folder_customized' displays where the JaWE2Openflow instance is present.



- Click on the Workflow_folder_customized folder to see a list of items like python scripts, dtml methods, page templates, acl_users folder and the OpenFlow object named Workflow.



The following items were added for Workflow:

Scripts

Approve
Reject
SendAgain
SubmitNew
data_approved
data_rejected

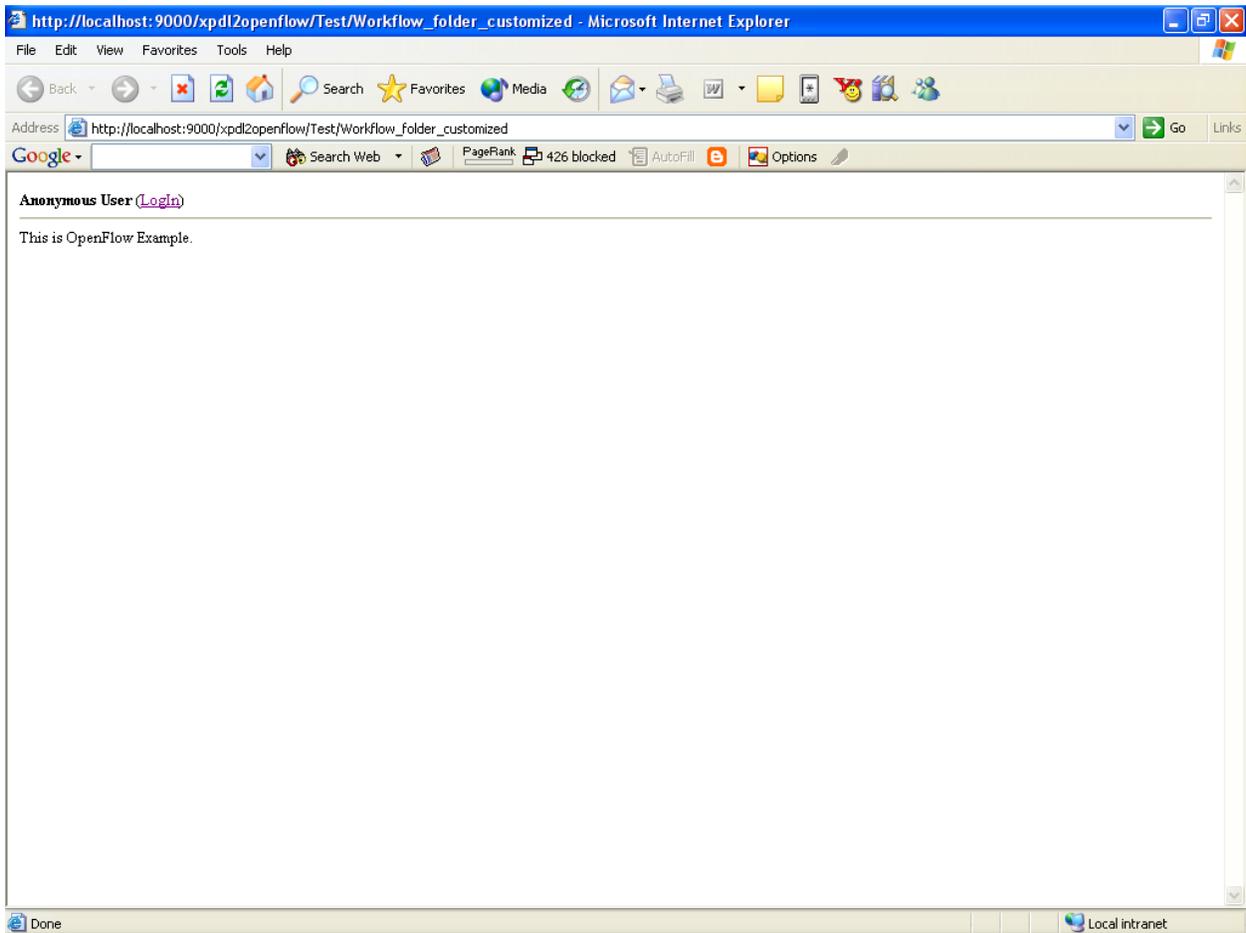
Page Templates

check_dataForm
doc_edit
index_html
submit_doc
worklist_html
standard_template

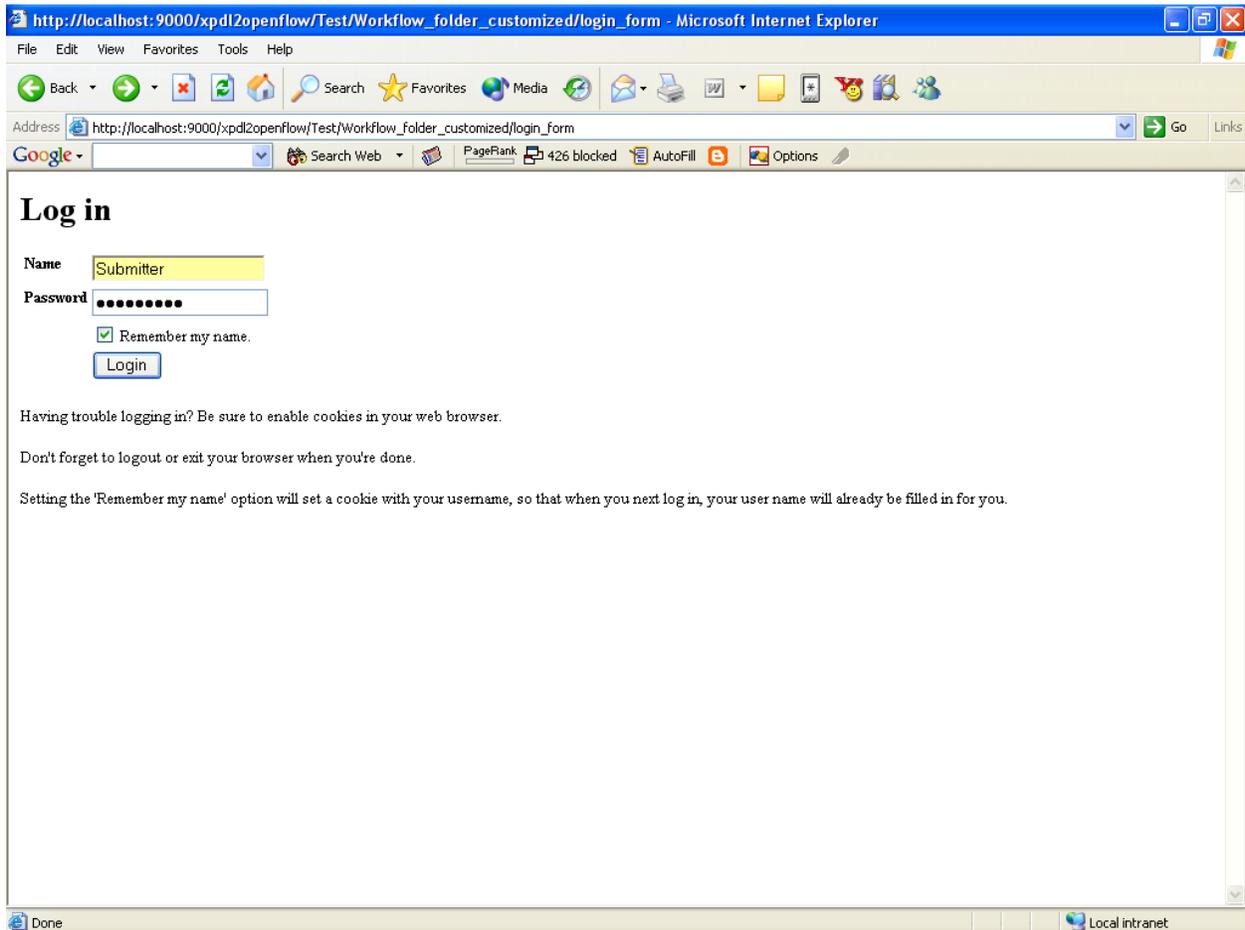
DTML Methods

login_form
logged_out
logged_in

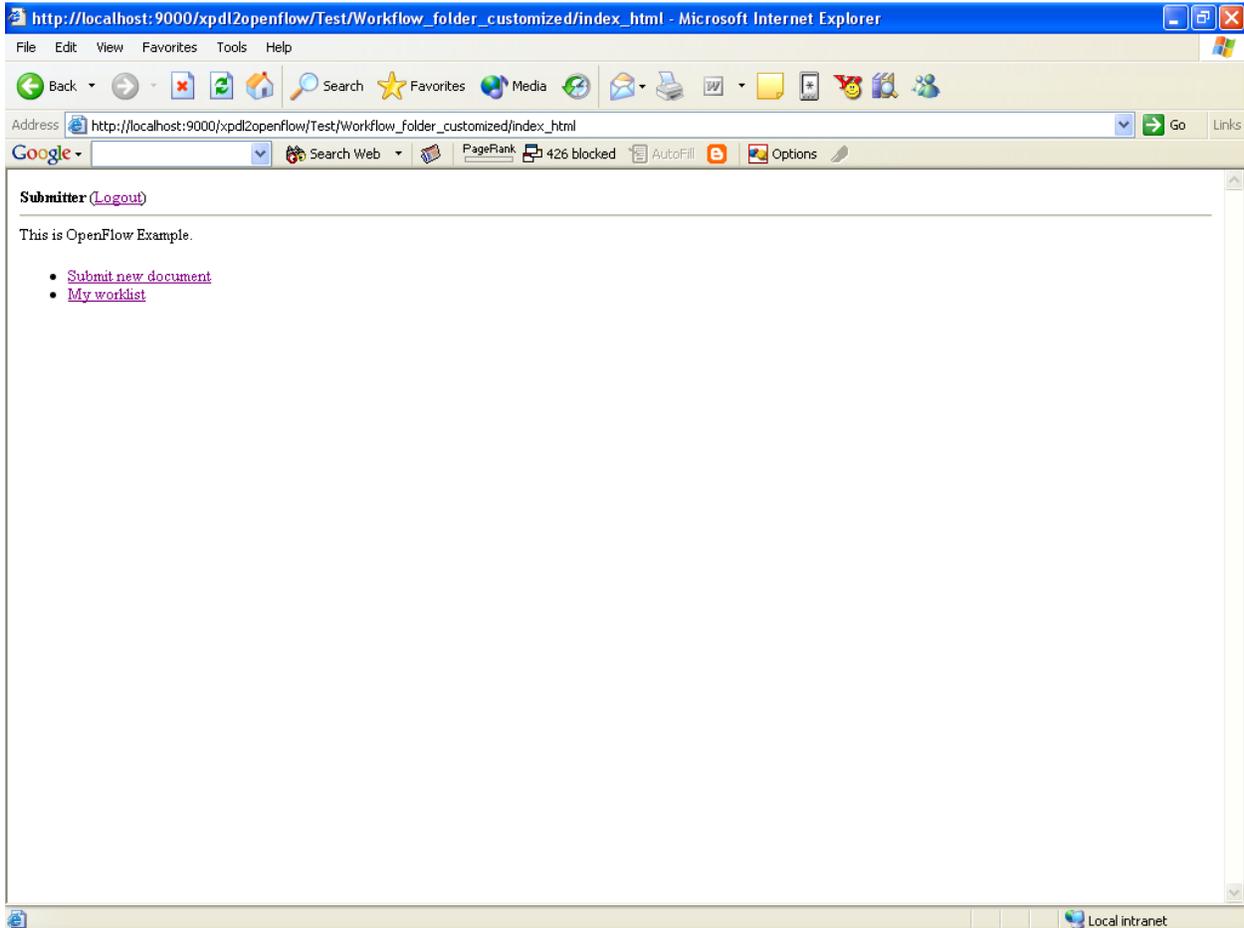
5. Open a browser and go to the URL where the Workflow_folder_customized.zexp file has been imported. For example, http://localhost:9000/xpd12openflow/Test/Workflow_folder_customized. A screen displays with an Anonymous User configured and a login link.



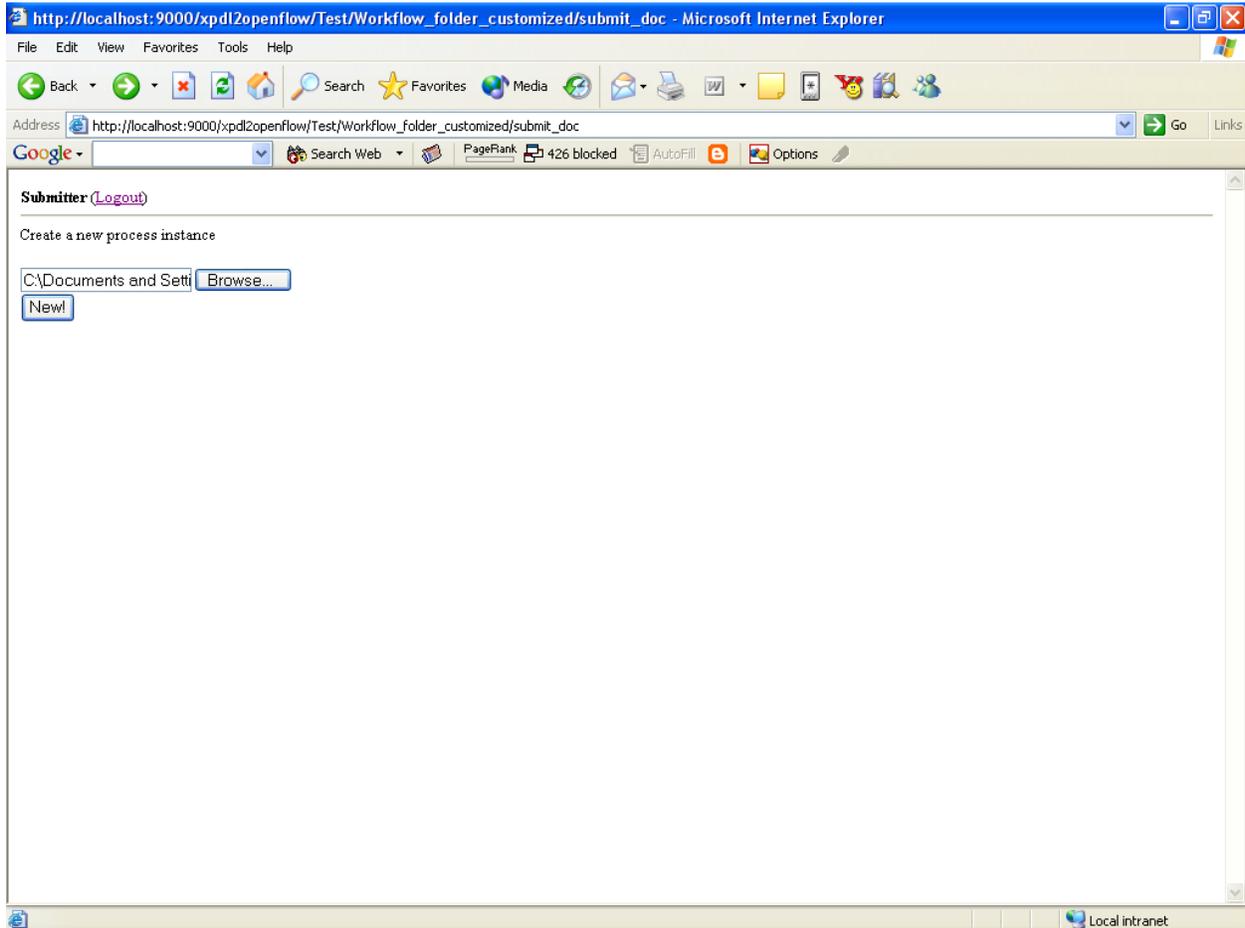
- Click on the Login link and login using “Submitter” as both the username and password. Click on the Login button.



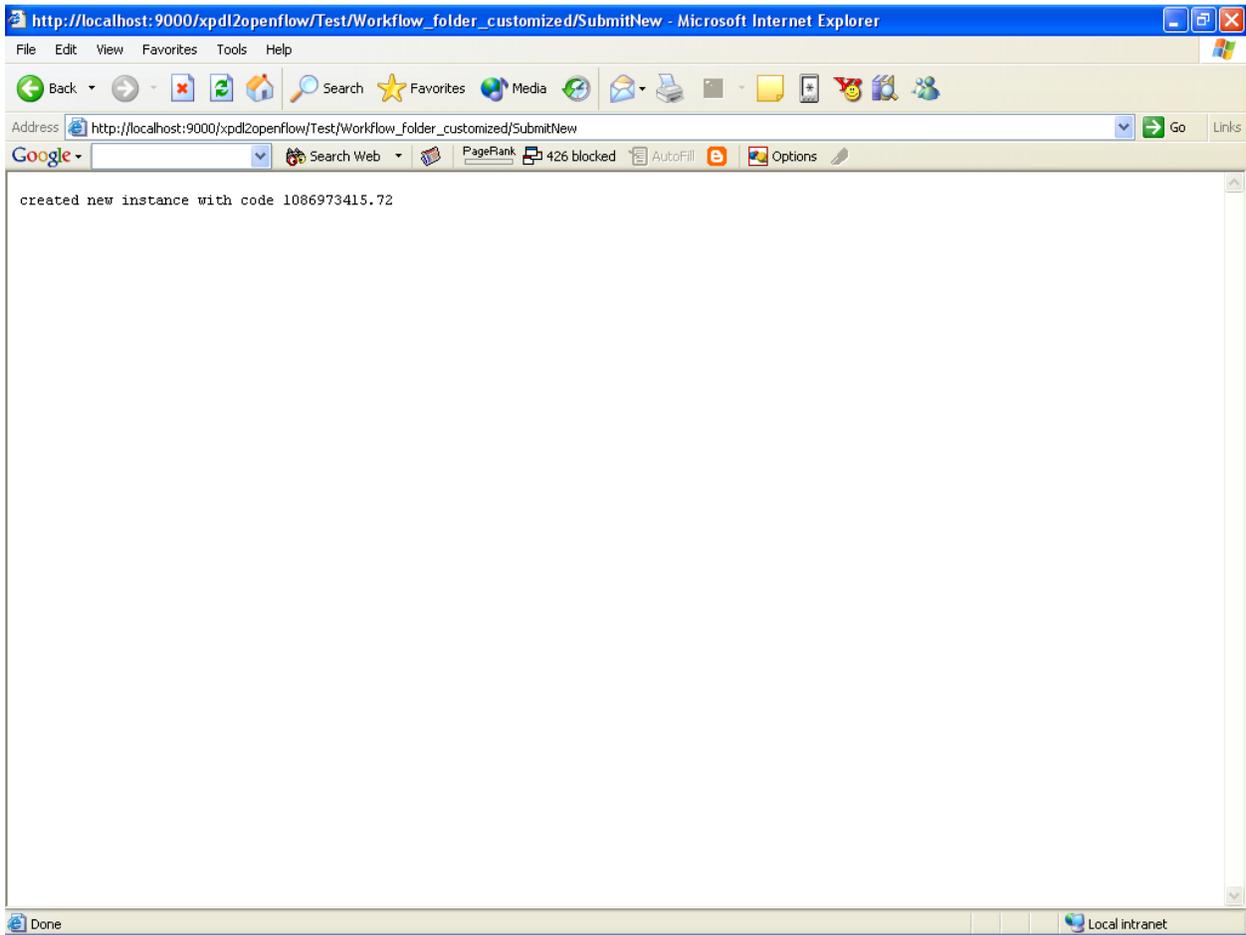
7. A screen displays showing the username, a logout link and links to Submit a new document and My worklist. Click on the Submit new document link.



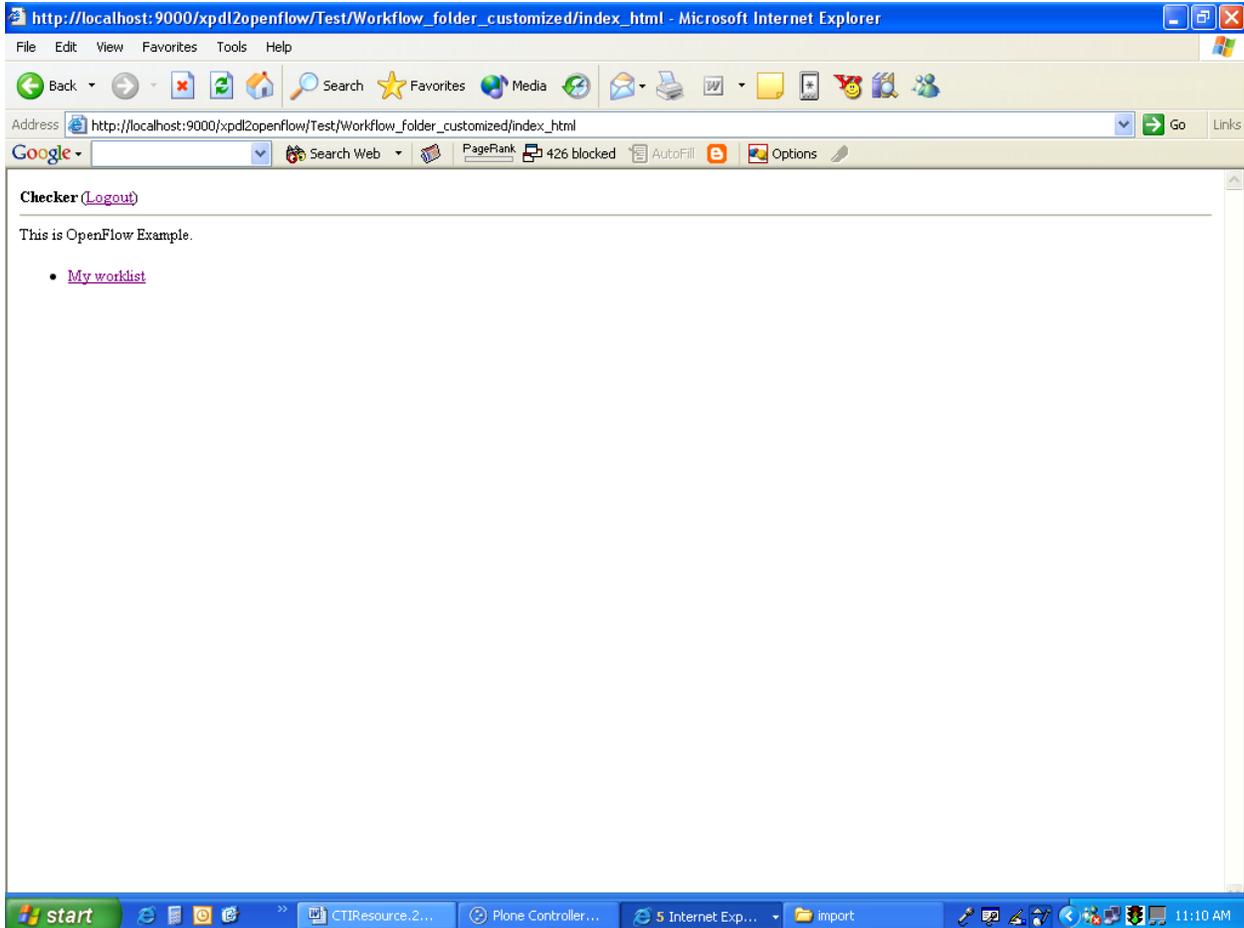
8. A screen for creating a new process instance displays. Click on the Browse button and select a document to upload then click on the New button.



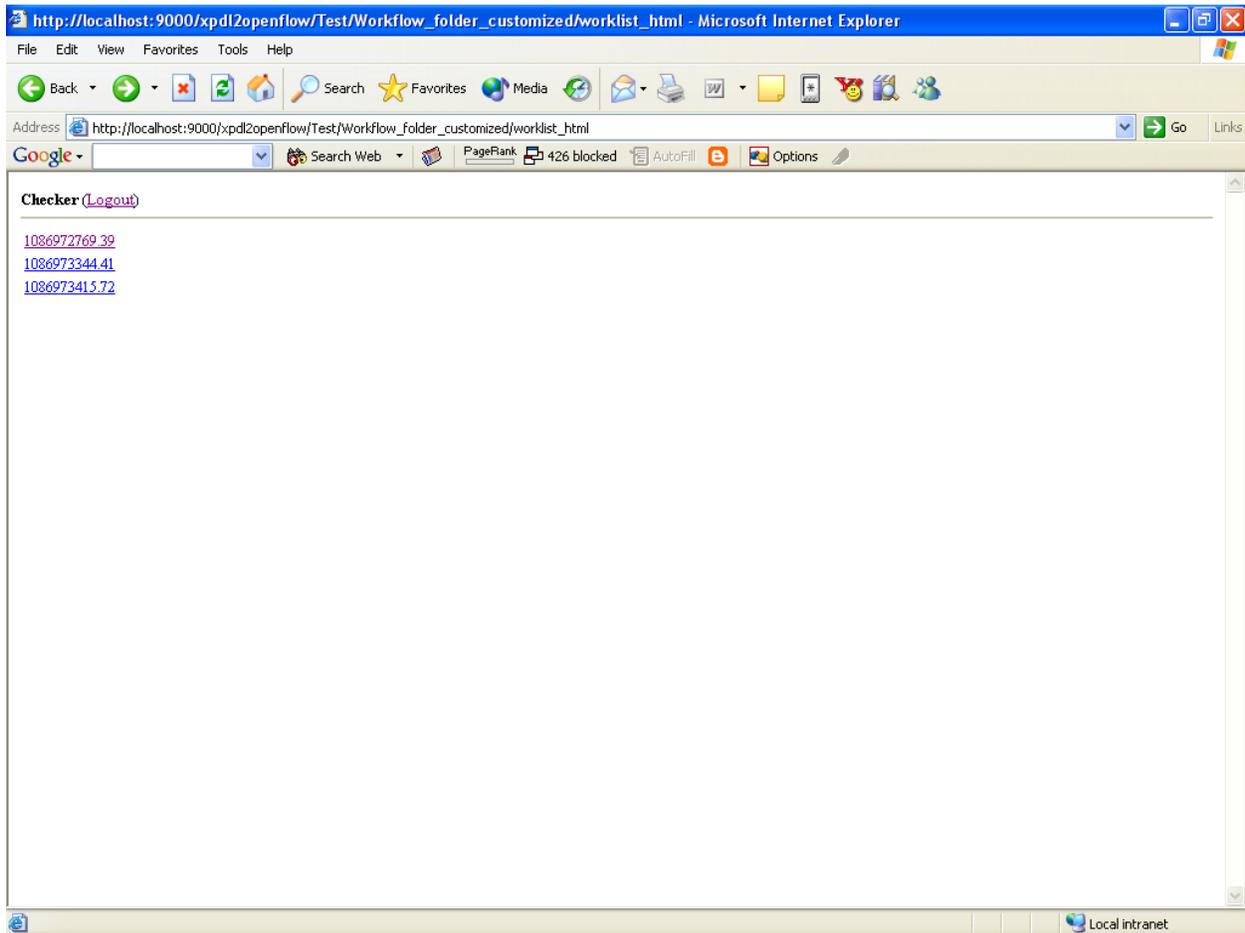
9. A screen displays that states that the workflow instance has been created.



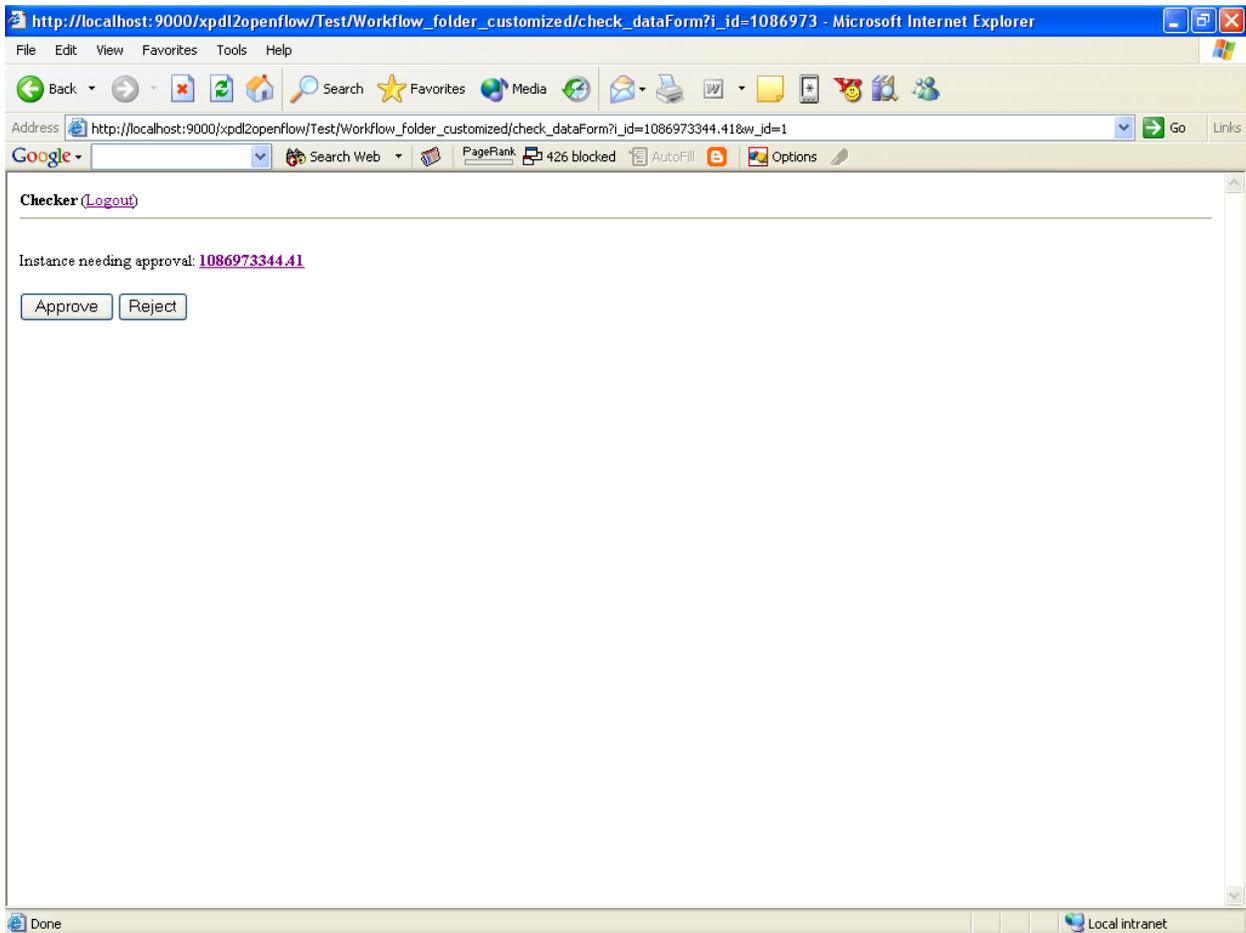
10. Open a browser and go to the following URL:
http://localhost:9000/xpd12openflow/Test/Workflow_folder_customized. Login using “Checker” as both username and password. A screen showing the user name at the top and a link to My worklist displays.



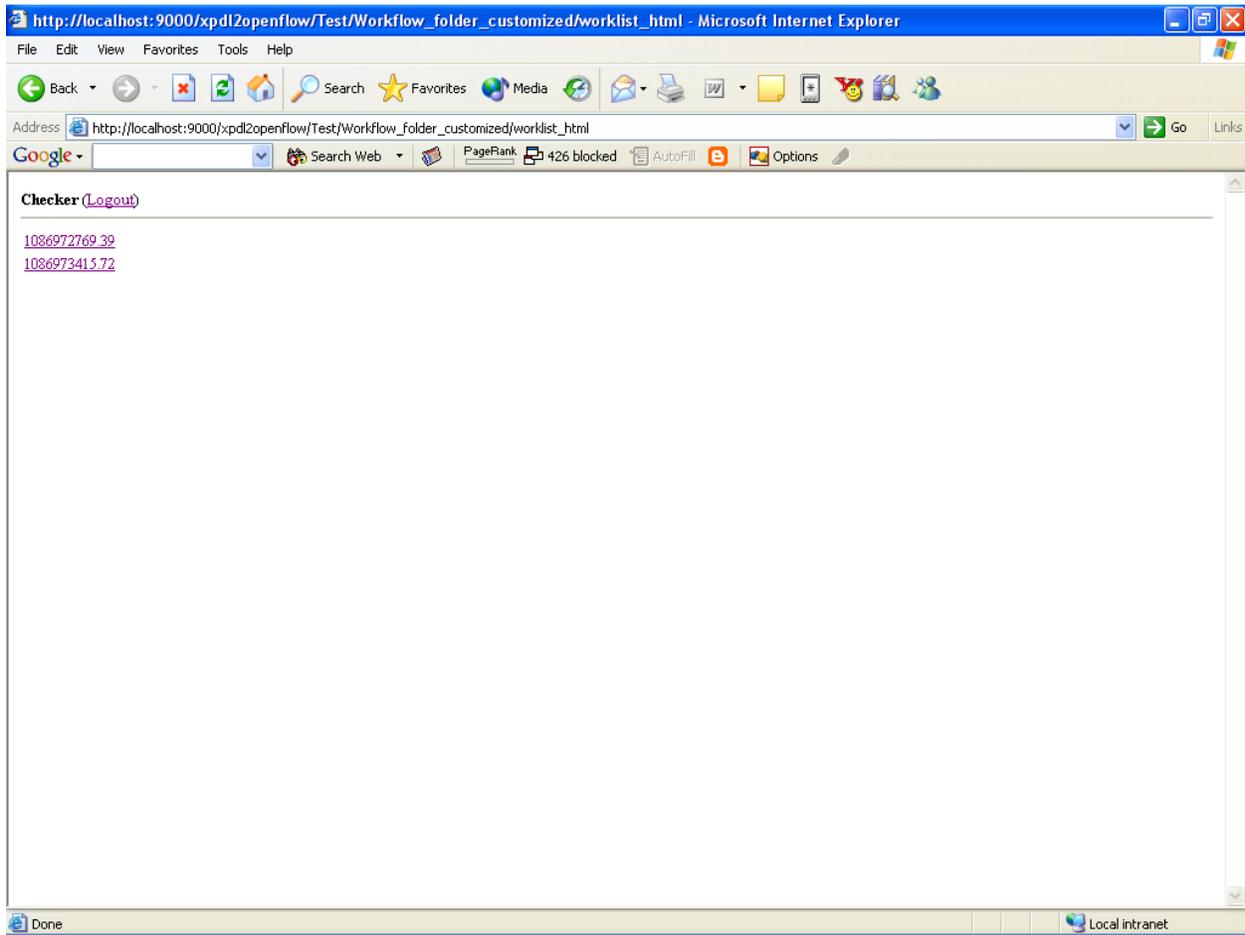
11. Click on the 'My worklist' link. The list of all instances pending approval displays.



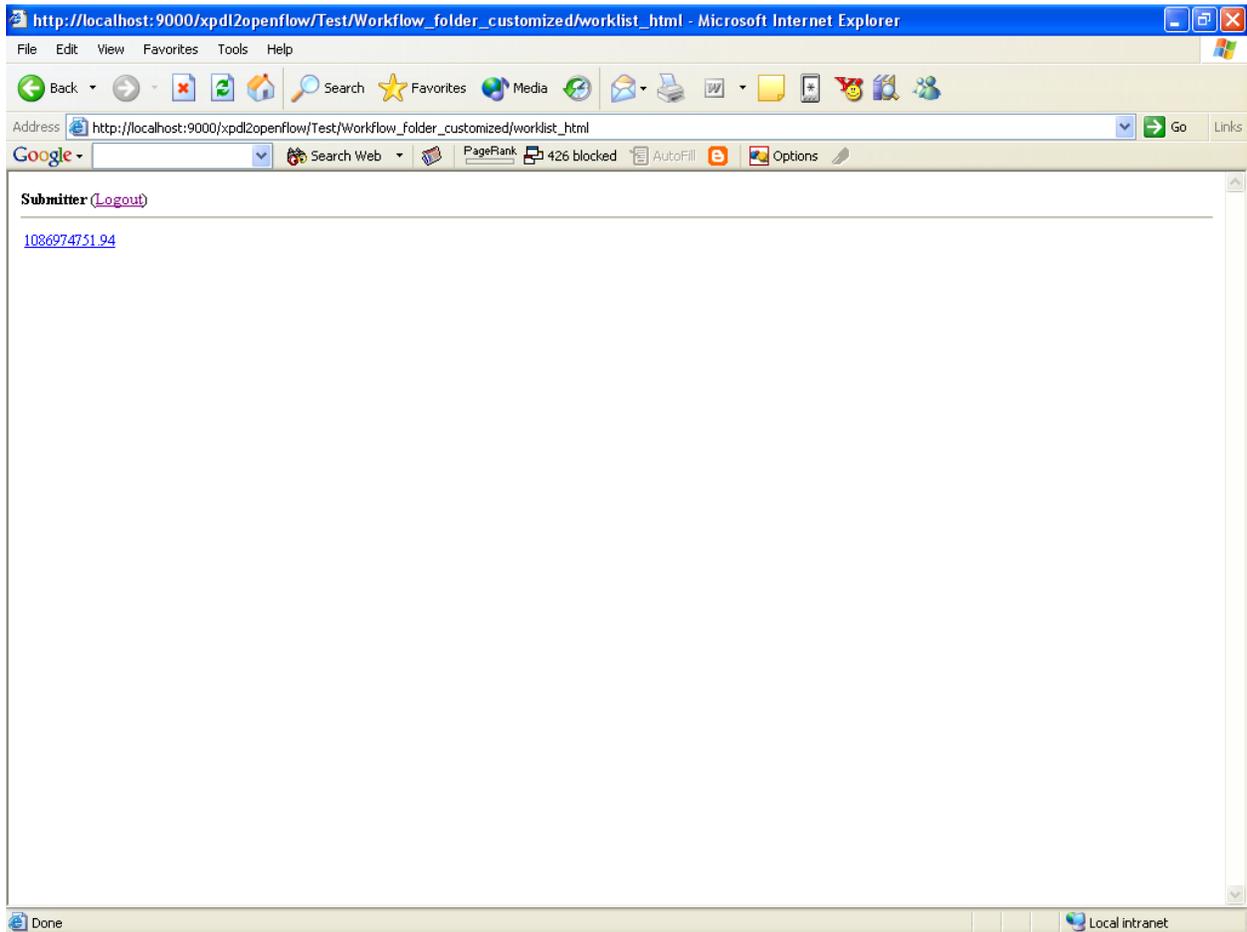
12. Click on an instance link, to see a screen stating that the instance needs approval with a link to the instance. Click on the link and it the uploaded file displays for review. Review it and click on the Approve button to approve it or reject it by clicking on the Reject button.



13. If approved, the document is removed from the worklist of the checker.



14. If a document is rejected, it is added to the worklist of the Submitter. The submitter can change it and submit it again.



8 Summary

8.1 Conclusion

JaWE2Openflow is a tool to convert a JaWE XPDL file to a Zope OpenFlow object.

8.2 Assumptions

The following assumptions were made while creating this product:

- Everything is defined in one XPDL file (i.e., applications, processes, participants, activities, transitions).
- All applications are defined in the <Applications> tag in the XPDL file. If the application type is “procedure” then it is considered to be a push application. If its type is “Application” it is considered to be an application.
- All Participants are defined in the <Participants> tag in the XPDL file and their type is “Role”.
- Workflow relevant data is defined at the package level and the data type is “Basic Type” only.
- Multiple processes can be defined in one XPDL file.

8.3 Limitations

The product has the following limitations:

- Currently, the product will not automatically convert the changes made to an XPDL file after the initial conversion. If changes are made to an XPDL file after it has been converted to an OpenFlow object, the corresponding changes to OpenFlow must be completed manually.
- Reverse workflow is not possible. If a Zope OpenFlow object is changed, the corresponding JaWE XPDL file must be changed manually.
- If an XPDL file is open in JaWE when it is uploaded in Zope (for conversion), a blank screen will display.