

Novell exteNd Director

5.0

www.novell.com

WORKFLOW GUIDE



Novell®

Legal Notices

Copyright © 2003 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher. This manual, and any portion thereof, may not be copied without the express written permission of Novell, Inc.

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Copyright © 2000, 2001, 2002, 2003 SilverStream Software, LLC. All rights reserved.

Title to the Software and its documentation, and patents, copyrights and all other property rights applicable thereto, shall at all times remain solely and exclusively with SilverStream and its licensors, and you shall not take any action inconsistent with such title. The Software is protected by copyright laws and international treaty provisions. You shall not remove any copyright notices or other proprietary notices from the Software or its documentation, and you must reproduce such notices on all copies or extracts of the Software or its documentation. You do not acquire any rights of ownership in the Software.

Patent pending.

Novell, Inc.
1800 South Novell Place
Provo, UT 85606

www.novell.com

exteNd Director *Workflow Guide*
December 2003

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

ConsoleOne is a registered trademark of Novell, Inc.

eDirectory is a trademark of Novell, Inc.

GroupWise is a registered trademark of Novell, Inc.

exteNd is a trademark of Novell, Inc.

exteNd Composer is a trademark of Novell, Inc.

exteNd Director is a trademark of Novell, Inc.

iChain is a registered trademark of Novell, Inc.

jBroker is a trademark of Novell, Inc.

NetWare is a registered trademark of Novell, Inc.

Novell is a registered trademark of Novell, Inc.

Novell eGuide is a trademark of Novell, Inc.

SilverStream Trademarks

SilverStream is a registered trademark of SilverStream Software, LLC.

Third-Party Trademarks

Acrobat, Adaptive Server, Adobe, AIX, Autonomy, BEA, Cloudscape, DRE, Dreamweaver, EJB, HP-UX, IBM, Informix, iPlanet, JASS, Java, JavaBeans, JavaMail, JavaServer Pages, JDBC, JNDI, JSP, J2EE, Linux, Macromedia, Microsoft, MySQL, Navigator, Netscape, Netscape Certificate Server, Netscape Directory Server, Oracle, PowerPoint, RSA, RSS, SPARC, SQL, SQL Server, Sun, Sybase, Symantec, UNIX, VeriSign, Windows, Windows NT

All third-party trademarks are the property of their respective owners.

Third-Party Software Legal Notices

The Apache Software License, Version 1.1

Copyright (c) 2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.
5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Autonomy

Copyright ©1996-2000 Autonomy, Inc.

Castor

Copyright 2000-2002 (C) Intalio Inc. All Rights Reserved.

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name "ExoLab" must not be used to endorse or promote products derived from this Software without prior written permission of Intalio Inc. For written permission, please contact info@exolab.org.
4. Products derived from this Software may not be called "Castor" nor may "Castor" appear in their names without prior written permission of Intalio Inc. Exolab, Castor and Intalio are trademarks of Intalio Inc.
5. Due credit should be given to the ExoLab Project (<http://www.exolab.org/>).

THIS SOFTWARE IS PROVIDED BY INTALIO AND CONTRIBUTORS ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTALIO OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Indiana University Extreme! Lab Software License

Version 1.1.1

Copyright (c) 2002 Extreme! Lab, Indiana University. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Indiana University Extreme! Lab (<http://www.extreme.indiana.edu/>)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Indiana University" and "Indiana University Extreme! Lab" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact <http://www.extreme.indiana.edu/>.
5. Products derived from this software may not use "Indiana University" name nor may "Indiana University" appear in their name, without prior written permission of the Indiana University.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS, COPYRIGHT HOLDERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JDOM.JAR

Copyright (C) 2000-2002 Brett McLaughlin & Jason Hunter. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.
3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jdom.org.
4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management (pm@jdom.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (<http://www.jdom.org/>)."

Alternatively, the acknowledgment may be graphical using the logos available at <http://www.jdom.org/images/logos>.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Phaos

This Software is derived in part from the SSLava™ Toolkit, which is Copyright ©1996-1998 by Phaos Technology Corporation. All Rights Reserved. Customer is prohibited from accessing the functionality of the Phaos software.

Sun

Sun Microsystems, Inc.

Sun, Sun Microsystems, the Sun Logo Sun, the Sun logo, Sun Microsystems, JavaBeans, Enterprise JavaBeans, JavaServer Pages, Java Naming and Directory Interface, JDK, JDBC, Java, HotJava, HotJava Views, Visual Java, Solaris, NEO, Joe, Netra, NFS, ONC, ONC+, OpenWindows, PC-NFS, SNM, SunNet Manager, Solaris sunburst design, Solstice, SunCore, SolarNet, SunWeb, Sun Workstation, The Network Is The Computer, ToolTalk, Ultra, Ultracomputing, Ultraserver, Where The Network Is Going, SunWorkShop, XView, Java WorkShop, the Java Coffee Cup logo, Visual Java, and NetBeans are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

W3C

W3C® SOFTWARE NOTICE AND LICENSE

This work (and included software, documentation such as READMEs, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications:

1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.
2. Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, the W3C Software Short Notice should be included (hypertext is preferred, text is permitted) within the body of any redistributed or derivative code.

3. Notice of any changes or modifications to the files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

Contents

About This Guide	11
PART I CONCEPTS	13
1 About Workflow in exteNd Director	15
What is workflow?	15
About the Workflow Modeler	17
About the workflow process	17
Workflow subsystem architecture	20
Workflow and pageflow	21
About the Workflow API	23
2 Designing Workflows	25
Getting started with workflow	25
Process design concepts	26
Single-branch flow	26
Multiple-branch flows	26
Pageflow in a workflow	28
Using the Workflow Return activity	29
Locking and unlocking workitems	30
3 Working with Activities	33
About workflow activities	34
Pageflow activity	35
User activity	37
Web Service activity	39
Rule activity	42
Java activity	43
Synchronize Merge activity	44
Start activity	45
Composer Service activity	46
Finish activity	48
4 Working with Links	49
About links	49
Simple link	50
Condition link	51
PART II TOOLS	53
5 Workflow Modeler	55
About the Workflow Modeler	56
Starting the Workflow Modeler	56
Process properties	58

About the Modeler window	59
Navigating, selecting, and moving objects	60
Adding activities	61
Workflow activity types	61
Adding links	63
Workflow link types	63
Drawing a link segment	64
Using scoped paths	64
Associating a scoped path with an activity	65
Copying scoped paths	67
Accessing scoped paths	69
Copying a scoped path to the clipboard	70
Creating link expressions	70
Validating a process	71
Adding text labels	71
Floating labels	71
Attached labels	72
Setting object display properties	73
Using the layout features	74
Full layout.	75
Incremental layout.	76
Setting preferences	77
Undoing a layout	77
Using the zoom features	78
Using the grid features	78
Using the Bird's Eye View	79
Creating a process resource view	80
6 Java Activity Wizard	81
About Java activities	81
Using the Java Activity Wizard	82
Coding the Java activity	84
Accessing a scoped path	84
7 Workflow Administration	85
Using the workflow administration portlets	85
Engine and Queue Administration Console	86
Workflow Administration Client	87
Auditing runtime processes	89
Generating runtime exception reports	89
PART III APPLICATIONS	91
8 Content Life Cycle Application	93
About the Content Life Cycle application	93
Running the application	94
Dynamic addressing of workitems	95
Content life Cycle application sources	96

PART IV REFERENCE	97
9 Workflow Tag Library	99
addressee	100
createProperty	100
forwardWorkitem	102
getDocument	102
getProcessList	103
getProperty	105
getPropertyList	106
getQueueStatus	108
getWorkitem	109
getWorklist	110
hasDocument	113
hasProperty	114
isDocumentLocked	115
isWorkitemLocked	116
setProperty	116
startProcess	118
updateDocument	119
workitemLock	121

About This Book

Purpose

This book introduces the basic concepts, architecture, and tools of the Novell® exteNd Director™ Workflow subsystem. It also presents some introductory workflow programming topics.

Audience

This book is for anyone who needs to design workflow processes or understand the features of the Workflow subsystem.

Prerequisites

This book assumes knowledge of Java programming and familiarity with the basic ideas of process automation.

I Concepts

- Chapter 1, “About Workflow in exteNd Director”
- Chapter 2, “Designing Workflows”
- Chapter 3, “Working with Activities”
- Chapter 4, “Working with Links”

1

About Workflow in exteNd Director

This chapter introduces the exteNd Director Workflow subsystem. It includes these topics:

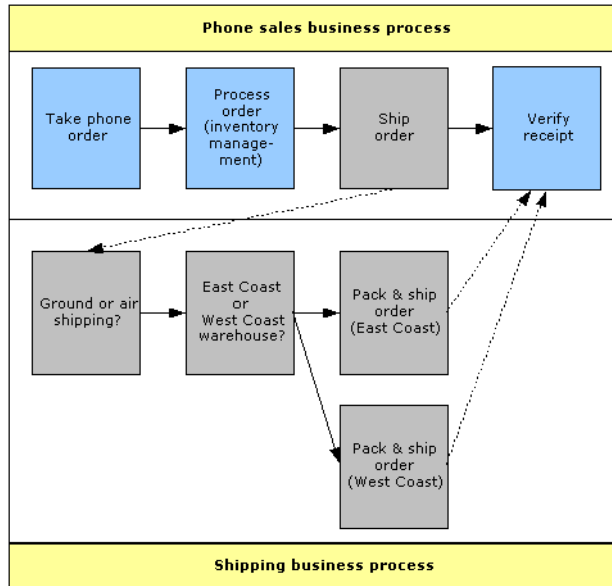
- ◆ [What is workflow?](#)
- ◆ [About the Workflow Modeler](#)
- ◆ [Workflow subsystem architecture](#)
- ◆ [Workflow and pageflow](#)
- ◆ [About the Workflow API](#)

What is workflow?

Workflow is the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules. Workflow can apply to business-to-customer, business-to-business, or internal processes.

Before you decide to implement workflow, your organization might be executing its work manually, with minimal automation if any. By studying the current work, a business analyst can isolate one or more procedures that together achieve an objective of the organization. Workflow designers call this set of procedures a *business process*.

The figure below shows two business processes: one that models how an organization executes telephone sales orders and one that models the shipping process.



A business process:

- ◆ **Has a definite starting point, or entry conditions**—In the telephone sales example, a customer phone call on a workday between 9 a.m. and 5 p.m. makes up a common set of entry conditions. Entry conditions trigger the start of a business process.
- ◆ **Accomplishes an objective of the organization**—Executing a sale is the objective of the business process triggered by the phone call.
- ◆ **Can be manual, partially automated, or completely automated**—In the shipping process, packing the product could be manual, choosing a carrier could be partially automated, and inventory management could be completely automated. Each step is still part of the business process.
- ◆ **Can be broken down into smaller business processes as necessary**—In the sales example, the original Ship Order step could be replaced by the shipping business process.
- ◆ **Has a definite ending point, or defined outputs**—For example, perhaps the business process cannot conclude until the customer verifies receipt.

Workflow automates the routing of work from activity to activity according to procedural rules. Work is expressed as a dynamic set of documents (data) associated with data that characterizes the set of documents (metadata). Taken together, the data and metadata are called a **workitem**. Users can act on the information in the workitem, accomplish tasks, update workitem information, and trigger the continuation of the workitem in the workflow.

The exteNd Director Workflow Subsystem provides the framework and resources for creating and running a workflow application using Java and XML technologies.

About the Workflow Modeler

The core of a workflow application is the *process definition*, an XML file that you create visually using the Workflow Modeler. A *workflow process* is the visual representation of the process descriptor in the Workflow Modeler.

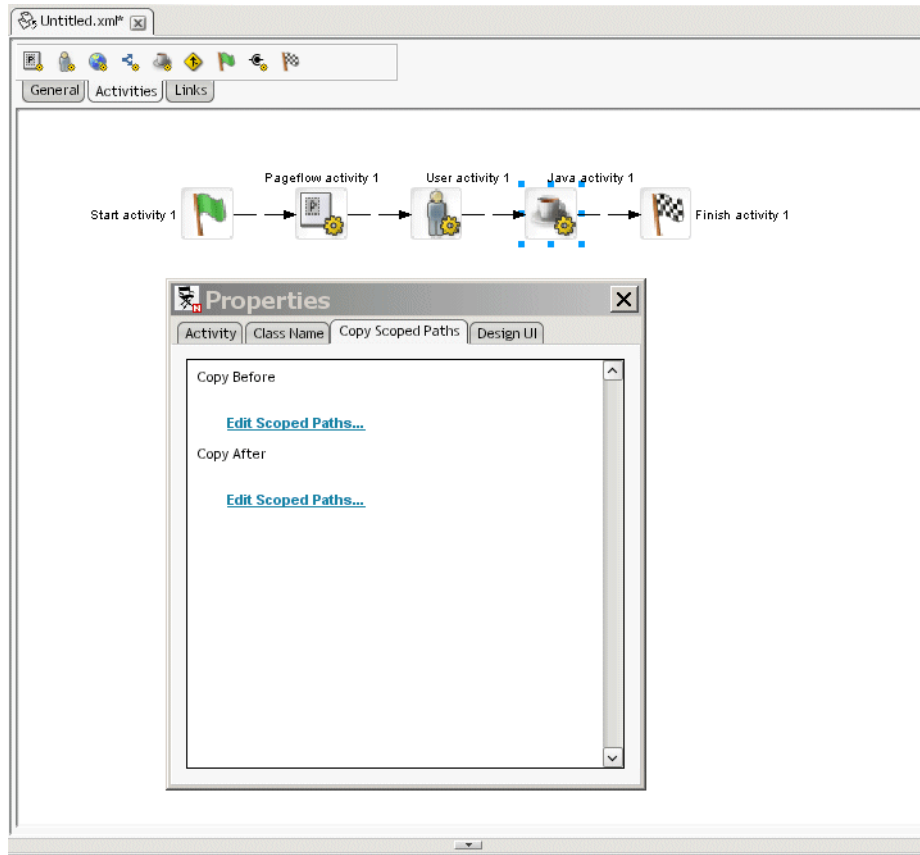
The *Workflow Modeler* that allows you to quickly and visually create a workflow process. The Workflow Modeler allows you to:

- ◆ Graphically lay out, annotate, and format your workflow
- ◆ Create, change, and delete activities and links
- ◆ Set activity and link properties
- ◆ Use the Scoped Path Navigator to associate activities and links with workitems

When you save a workflow process, the Workflow Modeler translates it into an XML-based file called a *process definition*. The process definition saves the layout and format of your workflow and translates the flow logic into a program the workflow engine can read and execute.

About the workflow process

Here is what a process looks like in the workflow Modeler:

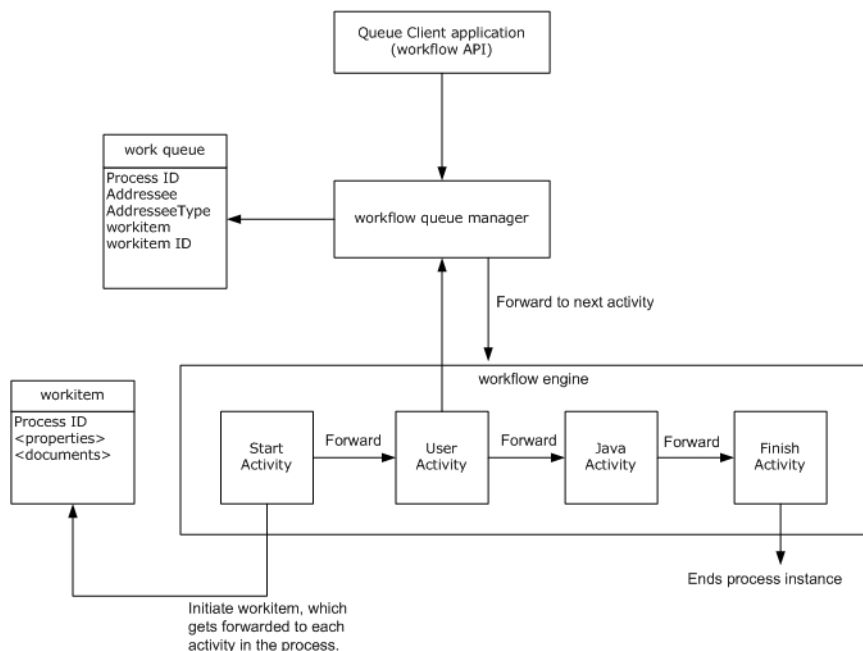


Here is a summary of the main features of a workflow:

Process component	Description
Activity	<p>Each process has a set of <i>activities</i> that control the runtime execution of the flow. The Workflow Modeler provides activities that present information to the user and respond to user interactions (User and Pageflow activities in the example above) and system activities that perform background tasks that are not visible to the user (Java activity in the example). Two of the system activities are required in every workflow:</p> <ul style="list-style-type: none">◆ Start activity: represents the Workflow subsystem functions necessary to create a new workitem and optionally to assign an initial document to the workitem.◆ Finish activity: represents the Workflow subsystem functions necessary to bring a workflow process to a normal end.
Link	<p>Links are what tie the activities in a workflow together (Links are represented by arrows in the workflow process). A <i>link</i> is a single logical path between two activities. An activity can have multiple source (incoming) links and multiple destination (outgoing) links. The Workflow Modeler provides two types of links:</p> <ul style="list-style-type: none">◆ Simple link: allows you to specify expressions that evaluate to true or false. When the expression for a simple link evaluates to true, the link is followed to the next activity.◆ Condition macro link: allows you to specify one or more conditions that are evaluated by the Rules subsystem. When a condition link evaluates to true, the link is followed to the next activity.
Scoped path	<p>Scoped paths is a core exteNd Director feature that provides a syntax for accessing application data. exteNd Director includes a scoped path called flow that lets you access workitem data from the Workflow modeler. The example above shows the Java activity accessing the copy scoped paths feature.</p>

Workflow subsystem architecture

The Workflow subsystem consists of two principal architectural components, the *workflow engine* and the *workflow queue*. These components interact with other workflow components, as shown in this diagram:



Workflow engine The workflow engine is responsible for executing and managing workflow processes. The diagram shows the execution of a single process instance. The workflow engine uses the process descriptor created with the Workflow Modeler to:

- ◆ Create new process instances
- ◆ Start, suspend, resume, and terminate activities
- ◆ Evaluate links and forward to the next activity
- ◆ Direct the workitem to the specified addressee

Workflow queue The workflow queue consists of the queue itself and a queue manager, which coordinates traffic between the workflow engine and workitems.

The queue persists workitems so they can be made available to workflow users. The queue stores workitem instances and data associated with them, such as the addressee (user) and the addressee type (user or security role). When the activity is completed, the queue manager tells the engine to forward the workitem to the next activity. System type activities that do not require user intervention (like Java activities) are not queued: They are executed in line by the workflow engine.

The diagram illustrates a single process instance, but in a real application there are likely to be many processes instances running at any given time. The queue can contain up to the number of active process instances for an activity (assuming that a workitem has reached the activity in each process).

About workitems The workitem is a container for the application data. It represents the state of the data at any given point in a process instance. In a typical workflow scenario, the workflow engine moves a workitem from activity to activity according to the link logic. (The workitem should not be confused with the data itself, which is typically modified by each activity.)

About the workflow client The workflow client provides access to the engine, queue and workitems through the workflow API. The diagram shows how a workflow queue delegate (instance of EbiQueueDelegate) accesses queued items for a specified addressee.

NOTE: exteNd Director includes core portlets for starting the workflow engine and accessing the queue. For more information, see [Chapter 8, "Content Life Cycle Application"](#).

Workflow and pageflow

It might be helpful to understand workflow by contrasting it with the exteNd Director *pageflow* model. A workflow and a pageflow are modeled in a similar manner, but a workflow application has some distinguishing features, as shown in this comparison:

Workflow	Pageflow
Process-based Workflow is a linear business process that might span several days, or even weeks.	Session-based A pageflow is open-ended and the duration is controlled by a single user.
Definite starting point Some specific event or condition triggers a process instance. This can be a system-generated event, like a monthly notification, or user-generated.	Designed entry point The entry point is determined by the flow itself, and is typically not triggered by an external event.

Workflow	Pageflow
<p>Multiple users</p> <p>A workflow assumes multiple users are performing discrete tasks at each activity.</p>	<p>Single user</p> <p>A pageflow application is driven by a single user.</p>
<p>Persistent data</p> <p>Instance data must be stored outside the session so workitems can be passed to subsequent users.</p>	<p>Session data</p> <p>A pageflow relies principally on instance data stored in the current session, although the application can access persistent data.</p>
<p>Definite ending point</p> <p>Some specific event or condition ends the process instance.</p>	<p>User-controlled exit point</p> <p>The user chooses when to end the session, although some specific action could trigger a work flow.</p>

NOTE: Although workflow and pageflow are used to build different types of applications, the Workflow subsystem provides facilities for integrating them. See ["Pageflow in a workflow" on page 28](#).

About the Workflow API

The Workflow API lets clients access workitems and define application logic for Java activities and User Activities. You can also use the APIs to access the engine and queue. Most of the classes for developing business logic are contained in the `com.sssw.wf.api` and `com.sssw.wf.client` packages. Here are some key classes:

Interface	Description
EbiWorkitemDelegate	Used by clients to interact with workitems. It provides methods for accessing workitem properties and associated documents, and for locking and unlocking workitems for users or user sessions.
EbiContext	Provides access to the workflow context for process instances.
EbiDocumentManager	Provides access to documents and document properties.
EbiQueueDelegate	Provides operations for accessing, forwarding, and reassigning queued workitems.
EbiWorkflowEngineDelegate	Provides methods for starting a process instance.
EbiJavaActivity	Provides methods for coding a Java Activity.
EbiActivity	Provides methods for accessing the behavior of activities at runtime.

2


Designing Workflows

This chapter describes how to get started with workflow and introduces some workflow design concepts. It includes these topics:

- ◆ [Getting started with workflow](#)
- ◆ [Process design concepts](#)
- ◆ [Pageflow in a workflow](#)

Getting started with workflow

Before you begin to develop a workflow you might want to look at the installed Content Life Cycle application. This workflow follows a document through several activities, including document assignment, authoring, approval, and publishing to the Content Management subsystem. It also uses the installed WorkflowStartProcess portlet and WorkflowQueue portlet to start the process and queue workitems for User activity addressees.

 For information about running the application and accessing the sources, see [Chapter 8, “Content Life Cycle Application”](#).

You should get familiar with the Workflow Modeler by opening the Content Life Cycle process or by creating your own process. If you have used the exteNd Director Pageflow modeler, you will find the design environment is similar.

 For more information, see [Chapter 5, “Workflow Modeler”](#).

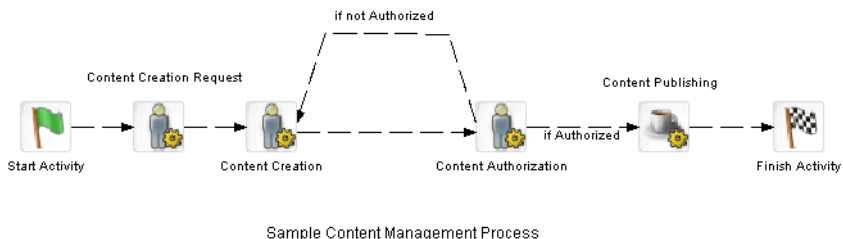
Process design concepts

The Workflow subsystem allows you to create two basic flow structures:

- ◆ Single-branch flow
- ◆ Multiple-branch flow

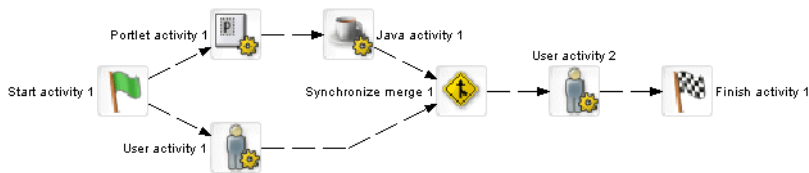
Single-branch flow

In a single-branch structure, the workflow engine forwards a single workitem from one activity to the next until the process finishes. The installed Content Life Cycle workflow is an example of a single-branch flow:



Multiple-branch flows

For more complex requirements, you can split the flow to allow for parallel or exclusive work by different process branches. In parallel execution the process follows two or more paths so that different work can be executed at the same time. For example, you might have two documents in the workitem that require different processing. Here is an example of parallel execution:



In this example, it is assumed that work is being forwarded on both branches following the Start activity. In the case of conditional branching like this, you must merge the branches using the **Synchronize Merge** activity.

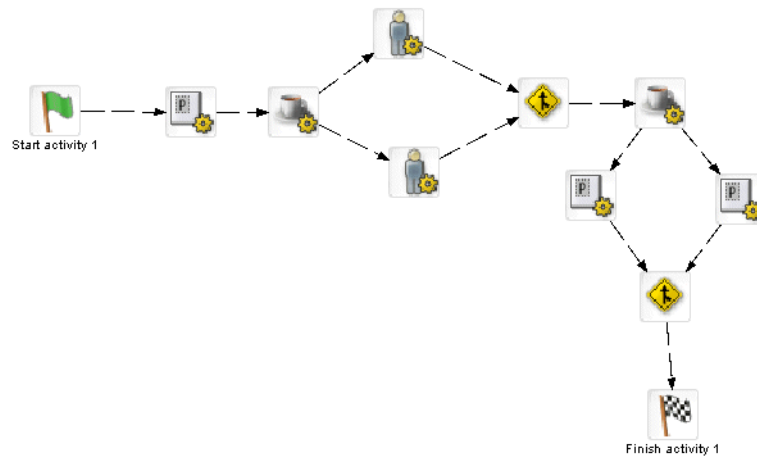
How synchronize merge works

Here is how the workflow engine synchronizes multiple branches:

- 1 When encountering a split, the engine forwards the workitem to the next activity in each branch, where the link evaluates to true.
- 2 The work is processed by each branch activity and forwarded according to the flow logic, until each one reaches the Synchronize Merge activity.
- 3 The workflow engine waits until work is completed on all branches and forwards the workitem to the next activity.

Multiple splits

This example illustrates how to handle multiple splits. You must merge each split before proceeding to another split:

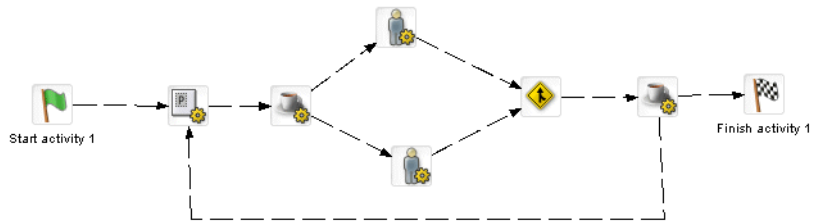


Exclusive branching and looping

The next example illustrates two more points:

- ◆ When your logic dictates that only one of multiple branches will execute exclusively, a Synchronize Merge activity is not necessary.
- ◆ You can loop back to a split that occurs earlier in the process as long as you return to an activity that is executed before the split.

In this example, it is assumed that the split after the last activity will proceed exclusively to the Finish activity or loop back to the first activity before the split:



 For more information, see [“Synchronize Merge activity”](#) on page 44.

Pageflow in a workflow

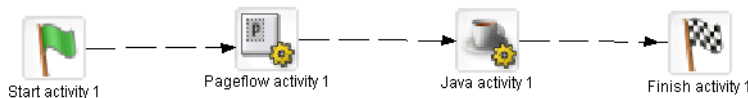
exteNd Director supports the integration of pageflows and workflows. The Pageflow activity in the Workflow Modeler allows you to specify a pageflow to run as a presentation activity within a workflow. This allows you to incorporate the rich client features of the exteNd Director Pageflow Modeler in your workflow process.

To use a pageflow in a workflow you need to implement these components:

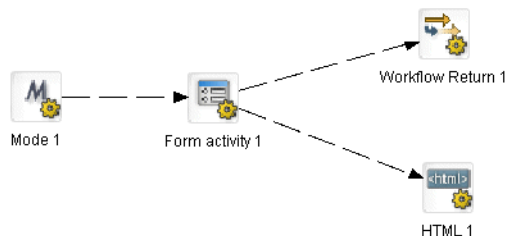
Workflow/Pageflow integration component	Description
Pageflow activity	Place a Pageflow activity in the workflow process where you want the pageflow to run. This activity passes the persistent workitem to the pageflow.
Workflow Return activity	Place a Workflow Return activity in the pageflow where you want to return the workitem to the workflow. This activity tells the workflow to forward the workitem to the next activity.
Flow scoped path	Use the Flow scoped path in both the pageflow and the workflow to store workitem instances. A pageflow that is included in a workflow accesses the same persistent workitem through the Flow scope.

Using the Workflow Return activity

Here is what a Pageflow activity looks like in a workflow process:



Here is what a pageflow that the Pageflow activity points to might look like:



This example shows how you can handle a common requirement in a workflow: allow a user to edit a workitem with the option to forward or not forward but persist the current state of the workitem for another session. For example, one link might be followed when the user selects a **Save and Forward** button and the other when the user selects **Save without Forwarding**.

The Workflow Return activity handles the first requirement. To address the second requirement this example uses an HTML activity that does nothing more than close the window:

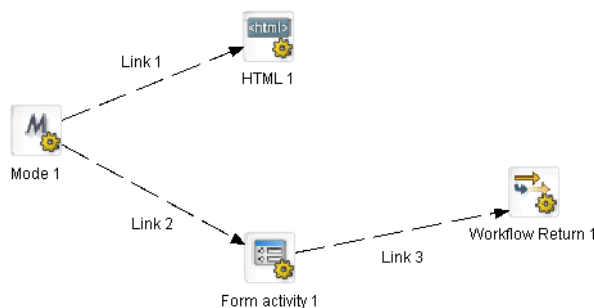
```
<script>self.close()</script>
```

Locking and unlocking workitem

Being able to lock and unlock a workitem for workflow users at runtime is a basic workflow requirement. The workflow API provides `lock()` and `unlock()` methods on `EbiWorkitemDelegate` for this purpose. For pageflows running in a workflow, there are some Flow scope properties you can use to lock and unlock workitems:

Flow scope property	Description
lock	Attempts to lock a workitem for the current user: <ul style="list-style-type: none">◆ Returns true if the workitem was successfully locked or if the workitem is already locked for this user.◆ Returns false if the workitem is already locked for another user.
unlock	Attempts to unlock a workitem for the current user: <ul style="list-style-type: none">◆ Returns true if the workitem was successfully unlocked or if the workitem is already unlocked.◆ Returns false if the workitem could not be unlocked because it is locked for another user.
persistent	Returns the state of the workitem. Returns true if the workitem is in a persistent state (workflow context) and false if not persistent (pageflow context).

The locking and unlocking of work takes place within the pageflow activity. As a rule, you want to lock the workitem when you start the pageflow, and unlock it before the Workflow Return activity is executed.



Suppose you have a Pageflow activity in a workflow calling the pageflow shown above, and you want to lock the workitem when the addressee accesses it from the queue. This is the basic procedure to follow:

NOTE: This section does not describe the mechanics of copying scoped paths in the Workflow Modeler. For more information, see [Chapter 5, “Workflow Modeler”](#). For more information about scoped paths, see the chapter on [using scoped paths](#) in *Developing exteNd Director Applications*

- 1 Use Flow/lock to lock the workitem and copy the return value to another location. For example:

```
Flow/lock ---> Flow/property/lockSuccess
```

You could do the copy scope before or after the Mode activity.

- 2 Test the return value and handle both cases.
 - ◆ If the return value is false, you could display a message that tells the user that the workitem is already locked by another user, as shown in Link 1 in the example
 - ◆ If the return value is true, provide the flow logic that lets the user update the workitem, as shown in link 2. When finished, link to a Workflow Return activity.
- 3 Use Flow/unlock to unlock the workitem before it is returned to the workflow.

For example:

```
Flow/unlock ---> Flow/property/unlockSuccess
```

You could do the copy scope before the Workflow Return activity.


In this case you might ignore the return value.

3

Working with Activities

This chapter describes the activities you can use in the Workflow Modeler. It includes these topics:

- ◆ [About workflow activities](#)
- ◆ [Pageflow activity](#)
- ◆ [User activity](#)
- ◆ [Web Service activity](#)
- ◆ [Rule activity](#)
- ◆ [Java activity](#)
- ◆ [Synchronize Merge activity](#)
- ◆ [Start activity](#)
- ◆ [Composer Service activity](#)
- ◆ [Finish activity](#)

 For information about using the Workflow Modeler, see [Chapter 5, “Workflow Modeler”](#).

About workflow activities


There are two types of workflow activities:

Category	Description	Activities
Presentation	Provide user interaction for the flow	Pageflow User
System	Perform background processing functions required by the flow	Start Web Service Rule Java Synchronize Merge Composer Finish

About activity properties The workflow activities have properties you can set in the Workflow Modeler. Some of these properties are common to all activities, and others are available only on some of the activities.

Some activities have a *primary property*. The primary property associates an object with the activity. For presentation activities, this is the item to display. For system properties, this is the object you want to execute.

About scoped paths *Scoped paths* are pointers to application data that you can select visually in the Workflow Modeler. You can use scoped paths to select the activity's primary property. In addition, most activities allow you to copy scoped paths before or after the execution of an activity.

 For more information, see the chapter on [working with scoped paths](#) in *Developing exteNd Director Applications*.

Pageflow activity



Description

The **Pageflow activity** represents a task or step in your business process that uses a portlet created using the Pageflow Modeler. Including a pageflow in a workflow allows you to take advantage of the Pageflow Modeler’s integrated client technology.

Properties

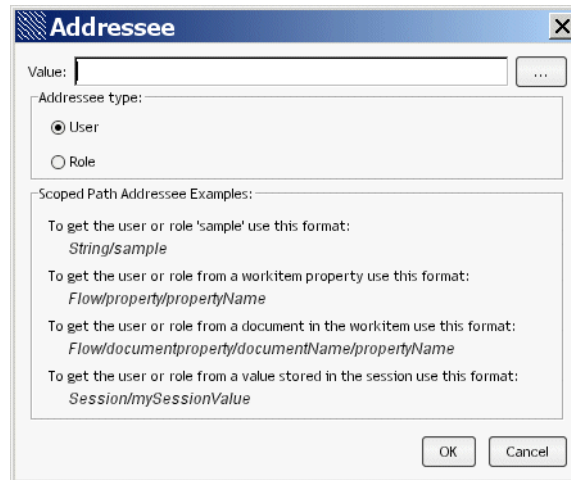
The properties for the Pageflow activity are

Property Inspector tab	Property name	Description
Activity	Name	A unique reference to this activity that is used for runtime processing.
	Description	A description of the activity intended for the Workflow Modeler user.
Pageflow Portlet	Portlet	Primary property. Specifies the pageflow descriptor in the resource set.
Addressees	Addressee	A valid user, group, or role name. See Specifying the addressee below.
Copy Scoped XPath	Copy Before	Copies data from one scoped path location to another before the activity is executed.
	Copy After	Copies data from one scoped path location to another after the activity is executed.
Design UI	Design UI properties control the design-time appearance of the activity. For more information, see “Setting object display properties” on page 73 .	

Specifying the addressee The addressee property determines where the workitem will be routed by the workflow engine. Each link can accept a single value for its Addressee property. Typically this value is a role, and the role defines a list of users with permission to perform work at the activity. You can also specify that the addressee value be dynamically derived at runtime from a workitem property.

➤ **To specify an addressee:**

- 1 Open the Property Inspector for the activity and choose the **Addressees** tab.
- 2 Click **Edit Addressee**:



- 3 In the **Value** field specify a scoped path for a user or role. See the examples at the bottom of the Addressee form.

- ◆ Click **User** if you are specifying a user known to your directory realm. For example:


`String/JSmith`

NOTE: If you are accessing an LDAP realm, specify the distinguished name. For example:

`cn=user,ou=users.`

- ◆ Click **Role** if you are specifying a security role descriptor. For example:

`String/workflowUser`

 For information about creating and using roles, see the chapter on [using security roles](#) in the *User Management Guide*.


Usage

At runtime, the Workflow engine delegates processing to the specified pageflow until the **Workflow-Return** activity in the pageflow is executed.

This activity ensures that the workflow engine does the following:

- ◆ Executes a forward operation to the next workflow activity in the containing workflow.
- ◆ Closes the portlet window, if the **close window** property on the **Workflow-Return** activity is selected.

This process is repeated for each included pageflow.

 For more information with examples, see [“Pageflow in a workflow” on page 28](#).

User activity





Description

The **User activity** represents a task or step in your business process that requires user interaction. Use this activity is for:

- ◆ A custom portlet, JSP page, or servlet not created or associated with a pageflow
- ◆ A non-Web client such as a word processing program or spreadsheet

Properties

The properties for the User activity are:

Property Inspector tab	Property name	Description
Activity	Name	Primary property. The name must match the name element in the activity policy descriptor , as described in “Binding activities to client types” on page 38 .
	Description	An optional description of the activity for the Workflow Modeler user.
Addressees	Addressee	A valid user or role name.  See “Specifying the addressee” on page 35 .
Copy Scoped XPath	Copy Before	Copies data from one scoped path location to another before the activity is executed.
	Copy After	Copies data from one scoped path location to another after the activity is executed.
Design UI	Design UI properties control the design-time appearance of the activity.  For more information, see “Setting object display properties” on page 73 .	

Binding activities to client types The default client for workflow applications is the Portal subsystem. The Workflow subsystem provides a mechanism for specifying different client types for workflow activities. For example, you might want the UI for a particular workitem to be a spreadsheet application, a Java client, or some other application.

With a User activity you are required to specify clients in a descriptor called **activity-policy.xml**.

Creating an Activity policy descriptor You need to create a descriptor that follows the schema is defined in **activitypolicy.xsd**, located in `library/WorkflowService/schema` in your installation directory. Place the descriptor in the `data/workflow-activity-policy` directory in your exteNd Director project's resource JAR.

NOTE: The name of the file must be `activity-policy.xml`

Here is the activity policy defined for three portlets used in the installed Content Life Cycle workflow application.

```
<?xml version="1.0" ?>
_ <activitypolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="activitypolicy.xsd">
_ <activity name="AssignContent">
<client type="portal" uri="/portal/portlet/WorkflowAssignPortlet">Portal</client>
</activity>
_ <activity name="CreateContent">
<client type="portal" uri="/portal/portlet/WorkflowCreatePortlet">Portal</client>
</activity>
_ <activity name="AuthorizeContent">
<client type="portal" uri="/portal/portlet/WorkflowAuthorizePortlet">Portal</client>
</activity>
</activitypolicy>
```

Element	Description
name	The name of the activity. This must match the Name property in the Workflow Modeler. For example, the Name property in the Workflow Modeler for the first activity would be AssignContent
client	The client type and URI for the application portlet or JSP page. The default client type is portal , which means the activity will use exteNd Director portal services to render the client. NOTE: You can also specify default as the client type and provide an implementation. You can also add a client tag under <code><activitypolicy></code> as a global default using the string default .


Adding client types To provide a different client type you must modify the activity policy file and add the new client type and associated URLs. The XSL of the WorkflowQueue core portlet (WorkflowQueueDefault.xsl) looks for **portal** as the client type and uses the associated URL as the link for the activity. To reference a different client type, modify the XSL for the queue portlet and the activity-policy client type accordingly.

NOTE: The WorkflowQueue portlet only supports direct URL references as an URI pointer.

Accessing policy information EbiWorkitemDelegate provides access to activity policy information. Here are some of the methods:

Client info method	Description
getClientTypes()	Returns an array of all client types for this activity
getClientURI()	Returns the URI of the specified client type
getPolicyDefaultURI()	Returns the default policy client URI for this activity
hasClientType()	Determines whether the activity includes the specified client type
hasClientDefault()	Determines whether the activity has a default client type

Coding the User activity You need to access the workitem and define your presentation logic in the implementation class. For code examples, refer to the sources for the User activities in the Content Life Cycle application.

 For more information see [“Content life Cycle application sources” on page 96](#).

Web Service activity



Description

The **Web Service activity** is a system activity that executes a Web Service.

The Web Service activity only provides support for document-style WSDL files that contain a schema. However, you can create a pageflow that uses an RPC-style Web Service by using a Java activity.

 For background information on Web Services, see the chapter on [Web Service basics](#) in *Utility Tools*.


 For details on how to use an RPC-style Web Service in the context of a pageflow, see the chapter on [working with RPC-Style Web Services](#) in the *Pageflow and Form Guide*.

Properties

The properties of the Web Service activity are:

Property Inspector tab	Property name	Description
Activity	Name	A unique reference to this activity that is used for runtime processing.
	Description	A description of the activity intended for the Workflow Modeler user.

Property Inspector tab	Property name	Description
Document Style	Web Service Input Document Path	Provides instance data for the request being made to the service. The input document is specified by means of a scoped path. Typically, the scope used for the input document is Flow/document.
	Web Service Name	The name of the service, as specified in the WSDL file.
	Web Service Operation	The name of the operation, as specified in the WSDL file.
	Web Service Output Parameter (optional)	The node name of the element returned by the service.
	Web Service Port Type	The port type for the service, as specified in the WSDL file.
	Web Service Return Document Path	Provides instance data for the response returned from the service. The output document is specified by means of a scoped path. Typically, the scope used for the output document is Flow/document.
	Web Service WSDL Document Path	Specifies the name of the WSDL file that describes the Web Service. TIP: Fill in this property first when you're working in the Workflow Modeler. Once the property sheet has a path to the WSDL file, it can automatically fill in many of the other properties associated with the activity.
Copy Scoped XPath	Copy Before	Copies data from one scoped path location to another before the activity is executed.
	Copy After	Copies data from one scoped path location to another after the activity is executed.

Property Inspector tab	Property name	Description
Design UI		Design UI properties control the design-time appearance of the activity.  For more information, see “Setting object display properties” on page 73

Rule activity




Description

The **Rule activity** represents the Rule subsystem functionality needed to execute a rule within a workflow. Rules allow you to build complex logic into your workflow by using installed conditions and actions.

Properties

The properties for the Rule activity are:

Property Inspector tab	Property name	Description
Activity	Name	A unique reference to this activity that is used for runtime processing.
	Description	A description of the activity intended for the Workflow Modeler user.
Rule	RuleID	Primary property. Specifies the rule descriptor in the resource set.
Copy Scoped XPath	Copy Before	Copies data from one scoped path location to another before the activity is executed.
	Copy After	Copies data from one scoped path location to another after the activity is executed.
Design UI		Design UI properties control the design-time appearance of the activity.  For more information, see “Setting object display properties” on page 73 .

Java activity



Description


The **Java activity** represents an unattended task or step in your process. This can be any kind of logic or processing that you want to happen automatically without user intervention. You specify a Java class you want to execute, which you create using the Java Activity Wizard. The class that you define extends [EbiJavaActivity](#).



For information about creating and coding the Java class, see [Chapter 6, “Java Activity Wizard”](#).

Properties

The properties for the Java activity are:

Property Inspector tab	Property name	Description
Activity	Name	A unique reference to this activity that is used for runtime processing.
	Description	A description of the activity intended for the Workflow Modeler user.
Class Name	Class Name	Primary property. The Java class that you want to execute.
Copy Scoped XPath	Copy Before	Copies data from one scoped path location to another before the activity is executed.
	Copy After	Copies data from one scoped path location to another after the activity is executed.
Design UI	Design UI properties control the design-time appearance of the activity.	 For more information, see “Setting object display properties” on page 73 .

Synchronize Merge activity



Description


The **Synchronize Merge activity** is used to synchronize a split in a workflow process. This activity is necessary only in cases when workitems are forwarded on more than one path of the split.

Here is how the workflow engine synchronizes multiple branches:


- 1 When encountering a split the engine forwards the workitem to the next activity in each branch where the link evaluates to true.
- 2 The work is processed by each branch activity and forwarded according to the flow logic, until each one reaches the Synchronize Merge activity.
- 3 The workflow engine waits until work is completed on all branches and forwards the workitem to the next activity.

Properties

The properties for the Synchronize Merge activity are:

Property Inspector tab	Property name	Description
Activity	Name	A unique reference to this activity that is used for runtime processing.
	Description	A description of the activity intended for the Workflow Modeler user.
Copy Scoped XPath	Copy Before	Copies data from one scoped path location to another before the activity is executed.
	Copy After	Copies data from one scoped path location to another after the activity is executed.
Design UI	Design UI properties control the design-time appearance of the activity.  For more information, see “Setting object display properties” on page 73.	

Usage

 For more information and examples, see [“Process design concepts” on page 26.](#)

Start activity




Description

The **Start activity** represents the Workflow subsystem functions necessary to create a new workitem, and optionally to assign an initial document to the workitem. The start activity is created when a new workflow process is created and cannot be deleted from the process. There is only one Start activity per process.

Properties

The properties for the Start activity are:

Property Inspector tab	Property name	Description
Activity	Name	A unique reference to this activity that is used for run-time processing.
	Description	A description of the activity intended for the Workflow Modeler user.
Document	Document Name	A unique reference for the start document that can be accessed from other flow objects.
	Start Document	A name and a string (separated by a space) that points to the contents of the start document. The pointer can be an URL.
Copy Scoped XPaths	Copy Before	Copies data from one scoped path location to another before the activity is executed.
	Copy After	Copies data from one scoped path location to another after the activity is executed.
Design UI	Design UI properties control the design-time appearance of the activity.  For more information, see “Setting object display properties” on page 73.	


Composer Service activity



Description

The **Composer Service activity** is a system activity that executes an exteNd Composer service. exteNd Composer services typically combine executable units of work called *components* and coordinate the flow of data between them. A typical service might include a series of components that receive an input XML document, perform sophisticated document mappings and transformations, collect information from back-end data sources, execute transactions on mainframes and AS/400s, process error conditions, send context-sensitive e-mail or JMS notifications, and/or return one or more XML response documents to the original requestor(s).


By breaking up a service's tasks into discrete components, important benefits—in terms of testing, debugging, code maintenance, and code reuse—can be realized.

 For details on using the Composer Pageflow Wizard, see the chapter on the [Composer Pageflow Wizard](#) in the *Pageflow and Form Guide*.

Properties

The properties of the Composer Service activity are:

Property Inspector tab	Property name	Description
Activity	Name	A unique reference to this activity that is used for runtime processing.
	Description	A description of the activity intended for the Workflow Modeler user.

Property Inspector tab	Property name	Description
Composer Service	Input Document	Provides instance data for the request being made to the service. The input document is specified by means of a scoped path. Typically, the scope used for the input document is Flow/document.
	Service	Indicates which service to run by specifying a service descriptor in the resource set
	Output Document	Provides instance data for the response returned from the service. The input document is specified by means of a scoped path. Typically, the scope used for the input document is Flow/document.
Copy Scoped XPath	Copy Before	Copies data from one scoped path location to another before the activity is executed.
	Copy After	Copies data from one scoped path location to another after the activity is executed.
Design UI	Design UI properties control the design-time appearance of the activity.  For more information, see “Setting object display properties” on page 73.	

Finish activity




Description

The **Finish activity** represents the Workflow subsystem functions necessary to bring a workflow process to a normal end. There must be one (and only one) Finish activity in a workflow process.

Properties

The properties for the Finish activity are:


Property Inspector tab	Property name	Description
Activity	Name	A unique reference to this activity that is used for runtime processing.
	Description	A description of the activity intended for the Workflow Modeler user.
Copy Scoped XPath	Copy Before	Copies data from one scoped path location to another before the activity is executed.
	Copy After	Copies data from one scoped path location to another after the activity is executed.
Design UI	Design UI properties control the design-time appearance of the activity.  For more information, see “Setting object display properties” on page 73 .	

4

Working with Links

This chapter describes the links you can use in a workflow process. It includes these topics:

- ◆ [About links](#)
- ◆ [Simple link](#)
- ◆ [Condition link](#)

 For information about adding links to your workflow, see [Chapter 5, “Workflow Modeler”](#).

About links


The following types of explicit links are supported within a workflow:

- ◆ Simple links
- ◆ Condition links

Links are evaluated in order of precedence When an activity is linked to more than one other activity, the workflow engine uses **precedence** to determine which link to follow. Each link has a precedence number associated with it. In cases where more than one link expression might evaluate to true, the engine evaluates the links in precedence order, following all links that return true.

What happens when all link expressions return false You can mark a link as the **default** path to follow out of a particular activity. When all of the link expressions evaluate to false, the workflow engine follows the default link.

About scoped paths *Scoped paths* are pointers to application data that you can select visually in the Workflow Modeler. You can use scoped paths to select test values for link expressions or to specify the path for an activity's primary property. You can also copy scoped paths after the execution of a link.

 For more information, see the chapter on [working with scoped paths](#) in *Developing exteNd Director Applications*.

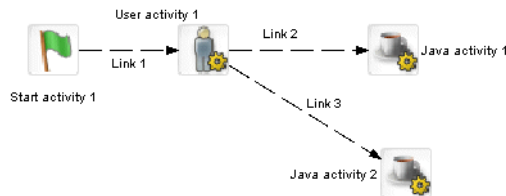
Simple link



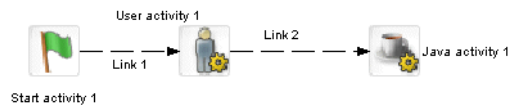
A **simple link** represents a simple path from one activity to another. Simple links allow you to specify expressions that evaluate to true or false. When the expression for a simple link evaluates to true, the link is followed to the next activity. When the expression evaluates to false, the path is not followed.

Simple links provide an easy way to choose between multiple target activities. When an activity finishes processing, you can use expressions on simple links to determine which activity should be executed next.




For example, you could use expressions on Link 2 and Link 3 in the following workflow to determine whether Java activity 1 or Java activity 2 would be executed after User activity 1:



If you do not specify an expression for a simple link, the workflow engine automatically moves to the target activity when the source activity finishes. For example, you would not need an expression on Link 2 in the following workflow, since the User activity1 activity has only one outgoing link:



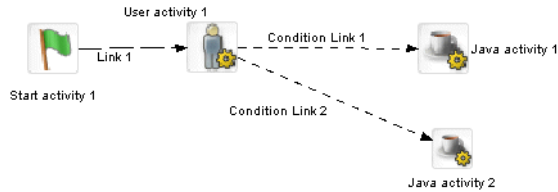
A Simple link has these properties:

Property Inspector tab	Property name	Description
Link	Name	A unique reference to this link that can be accessed in the flow.
	Description	A description of the link intended for the Workflow Modeler user.
	Copy Scoped Paths	Use to copy scoped data to another scope. The scope gets copied after the link is evaluated and before the target activity is executed.  For more information, see “Using scoped paths” on page 64.
	Default	Select to make this path the default. Use if you have multiple links to or from an activity. The default is the path that is used if no other path evaluates to true.
	Expression	Use to build a logical expression that evaluates to true or false. If you do not build an expression, the path will evaluate to true and will be executed.  For more information, see “Creating link expressions” on page 70.
	Precedence	Specify the order in which you want this link to be evaluated. Use if you have multiple links to or from an activity.
Design UI	Design UI properties control the design-time appearance of the link.  For more information, see “Setting object display properties” on page 73.	

Condition link





Like a simple link, a **condition link** represents a path from one activity to another. However, a condition link does not have an expression directly associated with it. Instead, it executes a condition macro created in the Rule Editor. A condition macro executes a set of conditions that contain reusable business logic:



Condition links evaluate to true or false. When a condition link evaluates to true, the link is followed to the next activity. When it evaluates to false, the path is not followed.

A Condition link has these properties:

Property Inspector tab	Property name	Description
Condition Link	Name	A unique reference to this link that can be accessed in the flow.
	Description	A description of the link intended for the WorkFlow Modeler user.
	Copy Scoped Paths	Use to copy scoped data to another scope. The scope gets copied after the link is evaluated and before the target activity is executed.  For more information, see “Using scoped paths” on page 64.
	Default	Select to make this path the default. Use if you have multiple links to or from an activity. The default is the path that is used if no other path evaluates to true.
	Condition macro ID	The name of the condition macro XML descriptor, as specified in the Rule Editor.  For more information, see the chapter on rule and macro editors in the <i>Rules Guide</i> .
	Precedence	Specify the order in which you want this link to be evaluated. Use if you have multiple links to or from an activity.
Design UI	Design UI properties control the design-time appearance of the link. For more information, see “Setting object display properties” on page 73.	

|| Tools

- Chapter 5, “Workflow Modeler”
- Chapter 6, “Java Activity Wizard”
- Chapter 7, “Workflow Administration”

5

Workflow Modeler

This chapter has general information about how to create workflow processes using the Workflow Modeler. It has these sections:

- ◆ *About the Workflow Modeler*
- ◆ *Starting the Workflow Modeler*
- ◆ *About the Modeler window*
- ◆ *Adding activities*
- ◆ *Adding links*
- ◆ *Using scoped paths*
- ◆ *Creating link expressions*
- ◆ *Validating a process*
- ◆ *Adding text labels*
- ◆ *Setting object display properties*
- ◆ *Using the layout features*
- ◆ *Using the zoom features*
- ◆ *Using the grid features*
- ◆ *Using the Bird's Eye View*
- ◆ *Creating a process resource view*

About the Workflow Modeler

The Workflow Modeler allows you to create and save an XML document that can be executed by the workflow engine at runtime. Creating a workflow process is a three-step process:

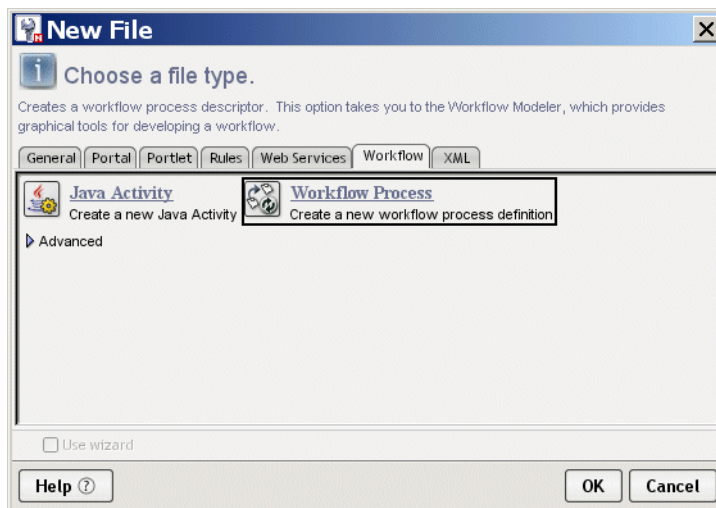
- 1 Use the graphing environment of the Workflow Modeler to lay out the logic of your process:
 - ◆ Add activities
 - ◆ Create links between activities
 - ◆ Add labels (optional)
- 2 Use the Property Inspector to configure activity and link properties that link the process to the resources of your exteNd Director environment.
- 3 Save the process in your project resource set. The Workflow Modeler converts it to a process definition that the workflow engine can execute.

Starting the Workflow Modeler

To start the Workflow Modeler you create a new process or open an existing one.

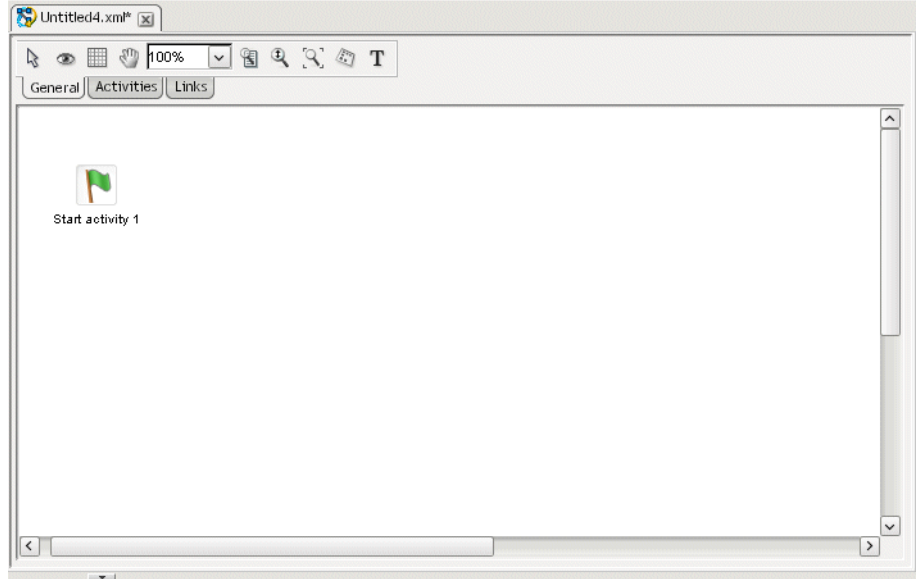
➤ **To create a workflow process:**

- 1 In the development environment, select **File>New**.
- 2 Click the **Workflow** tab:



- 3 Select **Workflow Process** and click **OK**.

The main window of the Workflow Modeler opens in the editing area. The flag icon represents the Start activity for the process:



➤ **To save a workflow process:**

- 1 Select **File>Save** from the exteNd Director menu.

If the Workflow Modeler detects any mistakes in the process, a popup message informs you and asks if you would like to save anyway. This validation occurs on all save events. You can address the warnings later.

- 2 Click **Yes**.

If this is your first save, the **Save As** dialog appears.

The contents of the workflow process definitions folder display in your [project resource set](#). Although you can specify another directory, it is recommended that you accept the default.

- 3 Specify a file name and click **Save**.

➤ **To open a workflow process:**

- 1 Select **File>Open** from the exteNd Director menu.
- 2 Navigate to the [workflow process](#) in your project resource set.
- 3 Double-click to open.

Process properties

You can access the process property sheet to view current information about the process and set UI properties.

➤ To access process properties:

1 Do one of the following:

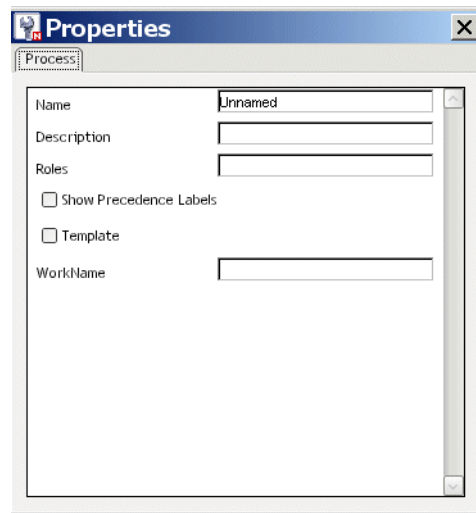
- ◆ Click the **Properties** icon on the Modeler toolbar:



OR


- ◆ With the cursor in the Modeler window, right-click and select **Properties**.

The Property Inspector displays the process properties:



2 Edit or view the process properties:

Property	Description
Name	The formal name of the process. The default is the name specified for the process descriptor.
Description	A description of the process intended for the Workflow Modeler user.

Property	Description
Roles	<p>A semicolon-delimited list of existing security roles that are authorized to start a process and create workitems. For example:</p> <pre>authors;developers;managers</pre> <p> For more information, see the chapter on using security roles in the <i>User Management Guide</i>.</p>
Show Precedence Labels	<p>Specifies whether precedence labels should be displayed in the workflow graph. When an activity is linked to more than one other activity, the Workflow engine uses precedence to determine which link to follow.</p> <p>Each link has a precedence number associated with it. In cases where more than one link expression might evaluate to true, the engine evaluates the links in precedence order, following the first link that returns true.</p>
Template	<p>Specifies whether this workflow can be used as a template for the creation of other workflows.</p>
WorkName	<p>A descriptive name for the Workflow Modeler user.</p>

About the Modeler window

Providing more screen space To provide more screen space you can close the Navigation and Output Panes in exteNd Director by clicking the panel view selectors:



Command features Here are the command features of the Workflow Modeler:



Navigating, selecting, and moving objects

➤ To navigate the window:

- 1 With the General tab selected, click the **Pan** button on the toolbar to change the cursor to pan mode:



- 2 Drag the **hand cursor** on the graph area to scroll the graph in the editing area. You can also use the horizontal and vertical scroll bars to achieve the same effect.

➤ To select objects:

- 1 With the General tab selected, click the **Select** button on the toolbar:



NOTE: Select mode is the default

- 2 Click any object to select it.
- 3 To select multiple objects, click empty space and drag the selection rectangle around the group.

➤ **To move objects:**

- 1 Select the object.
- 2 Drag it to where you want it.








When you move an activity, any associated links move with it.



Adding activities

An *activity* represents a *workitem* in the workflow process. You associate the activity with a workitem and/or business logic by setting properties on the activity icon, or node.

Workflow activity types

The following is a summary of the activities in the Workflow Modeler. For details about activity properties and usage, click on the appropriate item:

Node	Toolbar icon	Represents
Pageflow activity		A pageflow created with the Pageflow Modeler
User activity		A presentation activity for custom client implementations
Web Service activity		A system activity that executes a Web Service
Rule activity		A system activity that executes a business rule at runtime
Java activity		A system activity for executing custom business logic
Synchronize Merge activity		A system activity that provides a synchronization point for parallel branches of execution
Start activity		A system activity that marks the beginning of a workflow (required)

Node	Toolbar icon	Represents
Composer Service activity		A system activity that executes an exteNd Composer service
Finish activity		A system activity that marks the end of a workflow (required)

➤ **To add an activity:**

- 1 Click the **Activities** tab.

The activity toolbar displays:




- 2 Select an activity type.
- 3 Click anywhere on the graph to place the activity. Click and drag to move.

➤ **To delete an activity:**

- 1 Select the activity.
- 2 Press the **Delete** key or right-click and select **Delete**.

➤ **To access activity properties:**

- 1 Select the activity.
- 2 Right-click and select **Properties**—or click the **Eye** icon on the **General** tab.
- 3 Click the appropriate tab in the property sheet.



 The properties are specific for the activity type. For more information, see [“Workflow activity types” on page 61](#).

Adding links

A *link* is a single logical path from one activity to another activity. Each link represents a potential routing of workitems exiting a single activity. Links are represented by arrows in the Workflow Modeler.

Workflow link types

There are two types of links:

Link type	Toolbar icon	Represents
Simple link		A logical path from one activity to one or more other activities. On the link properties you specify: <ul style="list-style-type: none">◆ Routing logic using the link expression builder◆ Destination addressee as a user or role
Condition link		A logical path from one activity to one or more other activities. On the link properties you specify a Condition Macro that you created using the Rule Editor.

➤ To create a link:

- 1 Click the **Links** tab.

The link toolbar displays:



- 2 Select a link type.
- 3 Click on an activity and drag from the starting activity to the target activity, then release the mouse button.


 See also [“Drawing a link segment” on page 64](#).

➤ To delete a link:

- 1 Select the link.
- 2 Press the **Delete** key or right-click and select **Delete**.

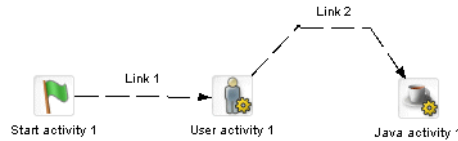
➤ To access link properties:

- 1 Select the link.
- 2 Click the **Eye** icon on the **General** tab—or right-click and select **Properties**.
- 3 Click the appropriate tab in the property sheet.

 Properties are specific for the link type. For more information, see [“Workflow link types” on page 63](#).

Drawing a link segment

A *segment* is the line between two points of a link. Segments are purely cosmetic. You can use them to enhance the legibility and appearance of the process model:




➤ To draw a segment:

- 1 Click any point on a link and drag in any direction.
The point you drag becomes the bend in the link.
- 2 Release the mouse button at the point where you want to end the segment.
- 3 Click again and drag to add another segment.
- 4 For the final segment, drag to the destination activity and click on it.

Using scoped paths

Scoped paths allow you to associate data with workflow activities and links. exteNd Director provides a set of predefined scoped paths that you can access using the Scoped Path Navigator.

 For background information about scoped paths, see the chapter on [working with scoped paths](#) in *Developing exteNd Director Applications*.

This section describes how to:

- ◆ Associate scoped paths directly with the primary property for an activity
- ◆ Copy scoped paths before or after an activity or a link executes in the flow

Associating a scoped path with an activity

You can use a scoped path to associate an object with an activity. To do this, you specify a scoped path as the value for the activity's primary property.

Activities you can associate You can do this with any of the following types of activities:

- ◆ Portlet activity
- ◆ User activity
- ◆ Java activity
- ◆ Rule activity
- ◆ Start activity
- ◆ Web Service activity
- ◆ Composer Service activity

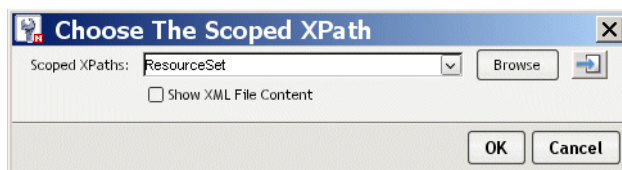
Two procedures Associating a scoped path with an activity requires two procedures:

- ◆ **To associate a scoped path with an activity's primary property:**
- ◆ **To specify a scoped path:**

➤ **To associate a scoped path with an activity's primary property:**

- 1 Select the activity, then right-click and select **Properties**.
- 2 In the Property Inspector, click the tab that has the primary property. This tab typically has the same name as the activity. For example, the Pageflow activity has a Pageflow tab where you can set the primary property.
- 3 Click the primary property for the selected activity. For details, see **“Workflow activity types” on page 61**.

The Choose The Scoped XPath dialog displays showing the default predefined path for this activity type:



The **Show XML File Content** check box applies to the Browse option. If selected, the XPath Navigator will display the tree contents of the XML file. If not selected the contents will not display. For more information, see **“To browse to a new path:” on page 66**.

- 4 Complete the next procedure to specify a scoped path.

➤ **To specify a scoped path:**

- 1 Use one of these methods to specify the path:

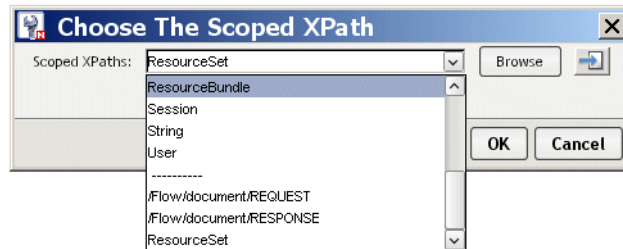
To enter the path directly:

- ◆ Type the path in the text box.

This requires that you know the syntax for the scoped path. For more information, see the section on [predefined scoped paths](#) in *Developing exteNd Director Applications*.

To select another path or a path already defined for this flow:

- ◆ Open the dropdown list for the current path:



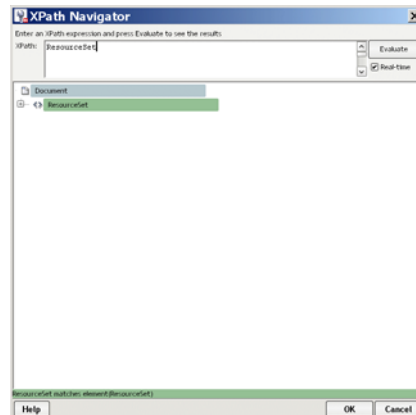
- ◆ Select another predefined path from the top of the list, or select a current path (already defined for this flow) from the bottom of the list.

TIP: If you choose a current path that accesses the resource set, you can display the contents of the file by clicking the icon next to the **Browse** button.

To browse to a new path:

- ◆ Click **Browse**.

When you click **Browse** the XPath Navigator displays:



The XPath Navigator is an interactive tool that allows you to specify the path using a tree view (where applicable) and the XPath expression builder. The available options depend on the scoped path you selected.



For more information on using the XPath Navigator, see the section on [using the XPath Navigator](#) in *Developing exteNd Director Applications*.

- 2 Click **OK**.

Editing a scoped path To edit an existing scoped path, repeat the two procedures above (**To associate a scoped path with an activity's primary property:** and **To specify a scoped path:**).

Copying scoped paths

With the exception of the Start and Finish activities, you can copy scoped paths before or after the execution of any activity, and copy scoped paths after the execution of a any link.

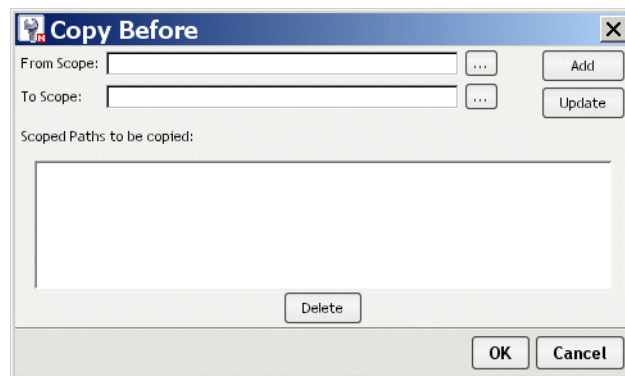


For design considerations that relate to copying scoped paths, see the section on [copying scoped paths](#) in *Developing exteNd Director Applications*.

➤ To copy a scoped path before or after an activity:

- 1 Select the activity, then right-click and select **Properties**.
- 2 In the Property Inspector click the **Copy Scoped XPath**s tab.
 - ◆ To copy the scoped path **before** execution, click the link **Edit Scoped XPath**s under **Copy Before**.
 - ◆ To copy the scoped path **after** execution, click the link **Edit Scoped XPath**s under **Copy After**.

The appropriate Copy dialog (Copy Before or Copy After) displays:



- 3 Click the ellipsis for **From scope**.

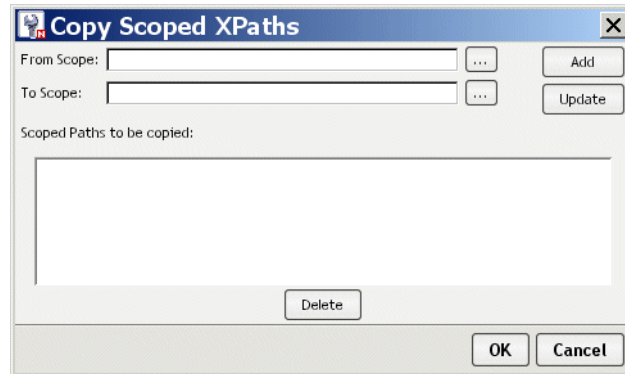
The dialog displays the default predefined scope for this activity type. If you want to use another scope, select it from the dropdown list.

- 4 To specify the path, use one of the methods described under **“To specify a scoped path:” on page 66.**
- 5 Click the ellipsis for **To scope**. Repeat the procedure for specifying the From scope.
- 6 On the Copy Scope dialog, click **Add**, then click **OK** to exit the dialog.
- 7 To edit an existing scope, repeat the procedure and click **Update** instead of Add.

➤ **To copy a scoped path after the execution of a link:**

- 1 Select the link, then right-click and select **Properties**.
- 2 In the Property Inspector click the link **Edit Scoped XPath** under **Copy Scoped XPath**.

The Copy Scoped XPath dialog displays:



- 3 Click the ellipsis for **From scope**
The dialog displays the default predefined scope for this activity type. If you want to use another scope, select it from the dropdown list.
- 4 To specify the path, use one of the methods described under **“To specify a scoped path:” on page 66.**
- 5 Click the ellipsis for **To scope**. Repeat the procedure for specifying the From scope.
- 6 On the Copy Scope dialog click **Add**, then click **OK** to exit the dialog.
- 7 To edit an existing scope, repeat the procedure and click **Update** instead of Add.

Accessing scoped paths

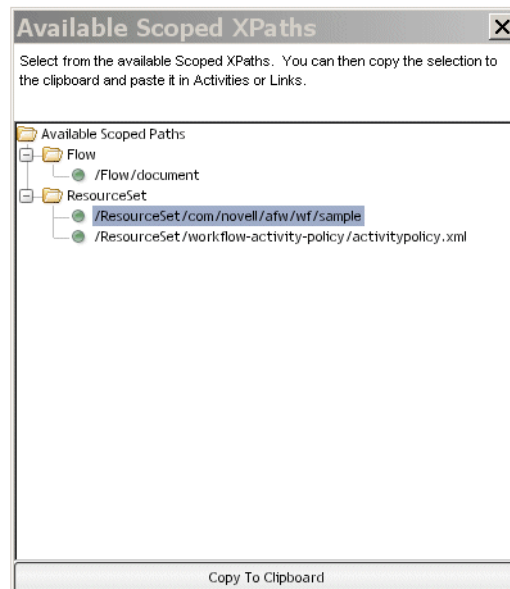
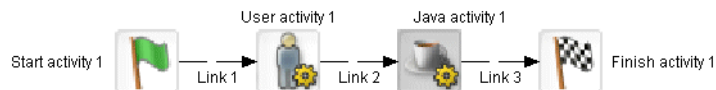
You can view all of the current paths used for a selected object in the flow or for all objects in the flow.

➤ **To access current scoped paths for an activity or link:**

- 1 Select the object.
- 2 Right-click and choose **Scoped paths**.
The Available Scoped XPath dialog displays.

➤ **To access all current paths in the flow:**

- 1 Do one of the following:
 - ◆ From the **Workflow** menu, select **View Scoped Paths**.
 - OR
 - ◆ With no objects selected, right-click and choose **Scoped paths**.The Available Scoped XPath dialog displays.
- 2 Click a scoped path. Notice that the objects that use it are highlighted, as shown with the Java activity in this example:



Copying a scoped path to the clipboard


Form the Available Scoped XPath dialog you can copy a selected path to the clipboard and paste it wherever you want.

➤ To copy a scoped path to the clipboard:

- 1 Access the scoped paths using one of the methods described in the preceding section.
- 2 Select the path you want to copy.
- 3 Double-click the path or click **Copy to Clipboard** at the bottom of the dialog.
- 4 Go to where you want to copy the path and press **Ctrl+V**.

Creating link expressions

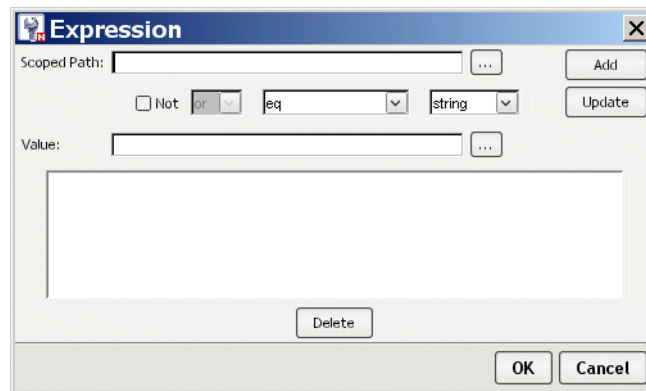
Link expressions let you dynamically route work based on runtime workitem values. The expression builder allows you to test workitem properties in simple or compound expressions that evaluate to true or false. Expressions apply to the Simple link only.

 For more information, see [“Simple link” on page 50](#).

➤ To specify a link expression:

- 1 Select a Simple link, then right click and select **Properties**.
- 2 In the Property Inspector select **Edit Expression**.

The Expression dialog displays:



- 3 Click **Choose Scope** to access a workitem.
- 4 To specify the path, use one of the methods described under [“To specify a scoped path:” on page 66](#).
- 5 In the **Value** field, specify the value to test against the scoped data.

- 6 Select the operator from the data operator dropdown list.
- 7 Select the data type (which must match the scoped data) in the data type dropdown.

The expression builder validates your entry:

- ◆ For **Boolean**, you must enter **true** or **false**.
- ◆ For **Timestamp**, use this format:

MM/DD/YYYY < optional: HH:MM:SS>

- 8 Click **Add** to add the expression to the text area at the bottom.
- 9 To add additional clauses to the expression, select the appropriate item from the logical operator dropdown, complete the clause, and click **Add** for as many clauses as you need.

NOTE: Enable the **Not** check box to negate the current expression.

- 10 To edit or delete an expression, click **Update** or **Delete**.
- 11 Click **OK** when you're done.

Validating a process

You can validate a process at design time whenever you choose. The Workflow Modeler analyzes the process structure and displays any errors encountered. Note that the validation applies to the design-time process structure only.

➤ To validate a process:

- ◆ Use one of these methods:
 - ◆ Select **Validate Process** from the Workflow toolbar menu.
 - OR
 - ◆ Right-click and select **Validate Process**.

Adding text labels

Labels are separate objects in the graphing environment and have their own property sheets. Labels have two forms: **floating** and **attached**.

Floating labels

Floating labels are simply text you place on your workflow graph and have no association with another workflow graph object. Titles, version numbers, notes, and legends are all good uses for floating labels.

➤ **To create a floating label:**

- 1 From the Workflow Modeler toolbar, select the **Label** button:



- 2 Click the location on the graph where you want the label to appear.
The label appears as a box with the text **Untitled** inside.
Each click creates a new label.
- 3 From the toolbar, select the arrow button.
- 4 Double-click inside a new label and edit the text.
- 5 Click outside the label to save your changes.

Attached labels

Most labels are associated with an activity or link and are called attached labels. By default, activities and links start with a label.

➤ **To edit an attached label:**

- 1 In the Workflow Modeler, right-click an activity icon or link.
- 2 From the popup menu, select **Create Label**.
A label appears directly below the activity or link.
- 3 Double-click inside the label and edit the text.
- 4 Click outside the label to save your changes.

An activity or link can have many labels. You can reposition the attached by dragging it to a new location. Note the line that appears as you drag. This line indicates the activity or link the label is attached to.

➤ **To format any label:**

- 1 In the Workflow Modeler, right-click the label and select **Properties**.
The Property Inspector displays showing the current formatting properties. (If the Property Inspector is already open, just click the label.)
- 2 Select the **Design UI** tab.
- 3 Make the changes you want.



For a description of the UI properties, see [“Setting object display properties” on page 73](#).

➤ **To delete a label:**

- 1 Select the label.
- 2 Press the **Delete** key.

Setting object display properties

Each activity, link, and label has a set of graphical properties associated with it. Select and right-click an element, and the properties of the element are displayed in the **Design UI** tab in the Property Inspector. The table below describes the graphical properties found on some or all of the elements:

Activity graphical properties	Description
Arrowhead Height	Customize the height (thickness) of the arrowhead for the selected links.
Arrowhead Width	Customize the width (length) of the arrowhead for the selected links.
Border Color	The color of the square outlining the activity. Click the color bar to display a standard color selection dialog.
Color	The background color of the activity. Click the color bar to display a standard color selection dialog.
Font	Click the data area to bring up a standard text formatting dialog.
Height	(Read-only) Height of the activity in pixels. You can enlarge or shrink the activity by dragging its handles.
Margin Height	For labels, the amount of space on the top and bottom between the text and the bounding box.
Margin Width	For labels, the amount of space on the left and right sides between the text and the bounding box.
Show Border	When enabled, displays a square outline around the activity, even when the background color is set to transparent.
Style	Choose a solid line or one of several dashed patterns from the dropdown list. Your choice affects only the currently selected link destination. To change several destinations at once, hold down the Shift key and click each one. Select a style from the Property Inspector.
Text color	The font color of the label. Click the color bar to display a standard color selection dialog.

Activity graphical properties	Description
Transparent	Overrides the color setting and makes the activity background transparent.
Width	(Read-only) Width of the activity in pixels. You can enlarge or shrink the activity by dragging its handles.
X Center	<p>When the workflow process is first created, the origin (0, 0) is the bottom-left corner of the graph. The graph automatically resizes in all directions as you create and drag items around. When this happens, the origin does not reset itself to the new bottom-left corner; it remains fixed.</p> <p>A positive value is the number of pixels above the origin the vertical center of the icon is currently located. A negative value indicates a position below the origin.</p> <p>Enter a new value to have the Workflow Modeler automatically move the activity to the vertical position specified.</p>
Y Center	<p>When the workflow process is first created, the origin (0, 0) is the bottom-left corner of the graph. The graph automatically resizes in all directions as you create and drag items around. When this happens, the origin does not reset itself to the new bottom-left corner; it remains fixed.</p> <p>A positive value is the number of pixels to the right of the origin the horizontal center of the icon is currently located. A negative value indicates a position to the left of the origin.</p> <p>Enter a new value to have the Workflow Modeler automatically move the activity to the horizontal position specified.</p>

Using the layout features

The *layout* is the arrangement of the activities, links, and labels in your graph. The Workflow Modeler has a sophisticated layout feature that can completely rearrange your graph to maximize readability and minimize space.

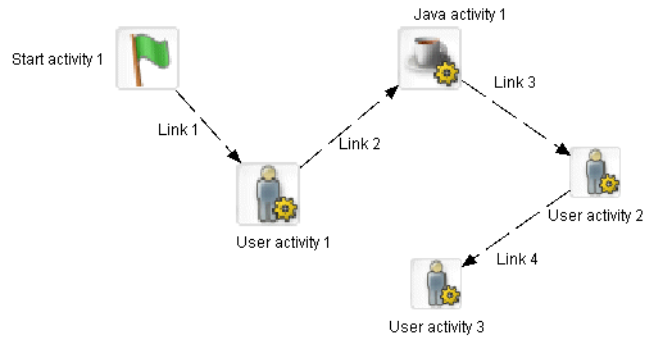
You can specify whether you want the new arrangement to have a horizontal or vertical orientation and whether you want links drawn as diagonal lines or composed of perpendicular segments.

There are two kinds of layout: *full layout* and *incremental layout*.

Full layout

Full layout gives the Workflow Modeler great freedom to move activities, links, and labels around the graph.

For example, this figure displays a hand-arranged layout:



This figure shows the result of applying a full layout to the hand-arranged layout:



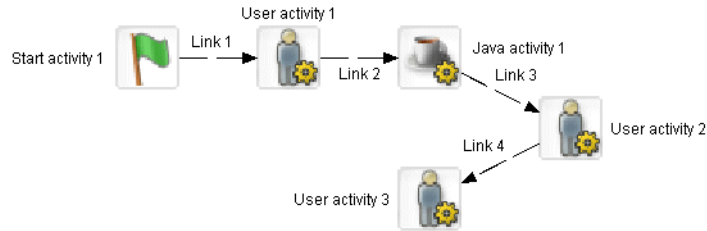
➤ To apply a full layout:

- ◆ Choose **Workflow >Full Layout**.
- OR
- ◆ Press **Ctrl+Shift+L**.

Incremental layout

Incremental layout attempts to make a graph more attractive and organized but also tries to keep the basic design of your hand-arranged layout.

This figure shows the result of applying an incremental layout to the hand-arranged layout:



➤ To apply an incremental layout:

- ◆ Choose **Workflow>Incremental Layout**.
- OR
- ◆ Press **Ctrl+Shift+M**.

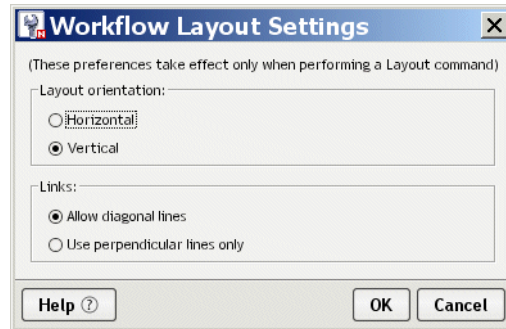
TIP: Try a layout on your graph and select **Edit>Undo** to back out of a layout you don't want.

Setting preferences

You can set layout preferences for a workflow graph. These preferences let you specify whether you want the new arrangement to have a horizontal or vertical orientation and whether you want link links drawn as diagonal lines or composed of perpendicular segments.

➤ To set layout preferences:

- 1 Choose **Workflow>Layout Settings**.



- 2 Select an orientation.
- 3 Select a link style.
- 4 Click **OK**.

Undoing a layout

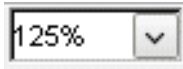



You can undo a layout selection and return to the previous layout.

➤ To undo or redo a layout:

- ◆ From the Main Menu select **Edit>Undo**.
- ◆ From the Main Menu select **Edit>Redo**.

Using the zoom features

The Workflow Modeler gives you four ways to zoom:

Zoom type	Toolbar icon	Description
Standard zoom		Allows you to pick from a list of common zoom percents
Marquee zoom		Allows you to drag and select a portion of the graph area to be zoomed to fill the graph window
Interactive zoom		Allows you to zoom up or down by dragging up or down on the graph
Fit in window		Allows you to zoom the graph window to show all the activities in the graph. To do this, it shrinks or enlarges the content of the current document to fit in the graph window.

➤ To zoom using the toolbar:

- ◆ Select the appropriate icon from the **General** tab on the toolbar.
OR
- ◆ Right-click in the Modeler window and select **Zoom**><option> from the popup window.

Using the grid features

The Workflow Modeler includes a drawing grid that works much like the grid in any graphics program:

- ◆ When the grid is visible, the corners of activity and label objects stick to the intersections of horizontal and vertical grid lines. This makes it much easier to line up objects and use consistent spacing.
- ◆ When the grid is invisible, objects can be positioned without constraint in one-pixel increments.
- ◆ Turning the grid on does not reposition existing items to align with the grid.

➤ **To turn the grid on or off:**

- ◆ Choose **Workflow>Toggle Grid**.
OR
- ◆ Click the **Grid** button from the Workflow Modeler toolbar:



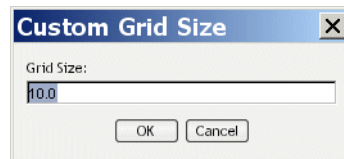
OR

- ◆ Press **Ctrl+Shift+G**.

➤ **To change the spacing of grid lines (grid size):**

- 1 Choose **Workflow>Grid Size**.

The Grid Size dialog displays:



- 2 Enter a value in pixels from 7.5 to 1000.
The default is 10.
- 3 Click **OK**.

Using the Bird's Eye View

The Bird's Eye View is a popup window that gives you a view of the entire workflow graph to help you find your way around in a large graph. The Bird's Eye View window:

- ◆ Appears when you click the Bird's Eye View button on the Workflow Modeler toolbar:



- ◆ Stays on top as you work in the graph area
- ◆ Is dismissed by clicking the X button in the upper-right corner
- ◆ Can be resized by dragging the corners
- ◆ Indicates the area visible in the graph window with a blue outline box

You can use the blue outline box of the Bird's Eye View window to do several useful things:

Doing this	Has this effect
Clicking outside the outline box	Centers the outline box on the point clicked and pans the graph area to correspond to the new location of the outline box
Dragging outside the outline box	Draws the outline box in the new location and pans the graph area to correspond to the new location of the outline box
Dragging inside the outline box	Pans the outline box and pans the graph area to correspond to the new location of the outline box
Dragging a corner of the outline box	Resizes the outline box and zooms the graph area to correspond to the new size of the outline box

Creating a process resource view

You can create a custom view of all resource files associated with a workflow. Once you've created the view, you can use display it on the **View** tab.

TIP: To get to the **View** tab, you need to first click on the **Resources** tab in the Navigation Pane.

- **To create a custom view for a workflow process:**
 - ◆ Choose **Workflow > Create Resource View**.

6

Java Activity Wizard

This chapter describes how to create a Java activity to use in a workflow process. It has these sections:

- ◆ [About Java activities](#)
- ◆ [Using the Java Activity Wizard](#)
- ◆ [Coding the Java activity](#)

About Java activities

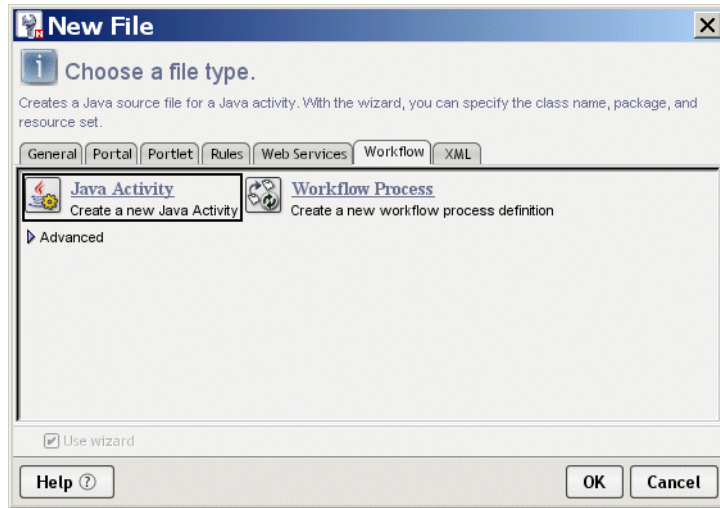
A Java activity is a Java class that executes within the context of an exteNd Director workflow application. A Java activity allows you to write custom business logic that executes automatically without user intervention.

You can create the Java activity using the Java Activity Wizard, code the resulting Java class template, and add the activity in the Workflow Modeler. The workflow engine automatically forwards work after the Java activity is processed.

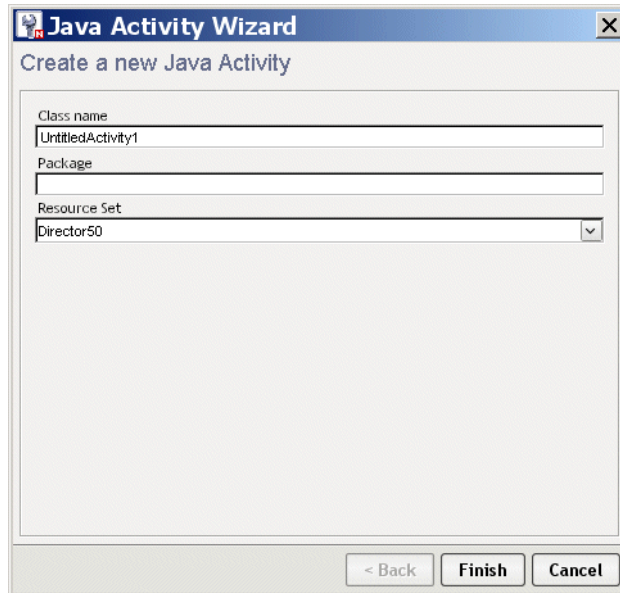
Using the Java Activity Wizard

➤ To generate the Java activity code template:

- 1 With your project open in exteNd Director, select **File>New>File**.
- 2 From the New File dialog, select the **Workflow** tab and click **Java Activity**:



- 3 Click **OK** to start the wizard. This panels displays:



- 4 Complete the wizard panel:

Option	What to do
Class name	Specify a class name for the Java activity
Package	(Optional) Specify a package hierarchy (with levels separated by periods) to place the Java activity in a subdirectory of the base directory. This affects only the directory where the Java activity is saved. For example, if the base directory is <i>ProjectDir/src</i> and you specify com.myco as the package, the Java activity will be created in <i>ProjectDir/src/com/myco</i> .
Resource Set	Select the Resource Set in which to store your application data.

- 5 Click **Finish**. The wizard generates the Java source template.
- 6 Click **OK** on the popup to access the template.

Coding the Java activity

The generated class implements the EbiJavaActivity interface and generates a method stub for the invoke() method. This method supplies the workflow context, and is called when work is forwarded to the Java activity in the workflow process.

Accessing a scoped path

The following example shows how to access a scoped path in the Java activity at runtime. This example uses the Session scope. Typically workitems in a workflow are stored in the Flow scope.

```
import com.sssw.wf.api.*;

public void invoke(EbiContext context) {

    try {
        // how to get a value from a scopedPath. ( assuming a request var of fname )
        com.sssw.fw.api.EbiScopedPath fname =
            com.sssw.fw.factory.EboScopedPathFactory.createScopedPath(
                "/Request/param/fname");
        String theFirstName = (String)fname.getValue( context );

        // how to set a value on a scopedPath.
        com.sssw.fw.api.EbiScopedPath sessionDoc =
            com.sssw.fw.factory.EboScopedPathFactory.createScopedPath(
                "/Session/DOC");
        sessionDoc.setValue( context, "mySessionDocValue" );

        // how to copy the request Referer into a session variable
        com.sssw.fw.api.EbiScopedPath from =
            com.sssw.fw.factory.EboScopedPathFactory.createScopedPath(
                "/Request/prop/Referer");
        com.sssw.fw.api.EbiScopedPath to =
            com.sssw.fw.factory.EboScopedPathFactory.createScopedPath(
                "/Session/Referer");
        com.sssw.fw.core.EboScopedPathUtil.copy( context, from, to );
    }
    catch (Exception e) {
        System.out.println(e);
    }
}
```

7

Workflow Administration

This chapter describes how to use the Workflow administration tools and features. It has these sections:


- ◆ [Using the workflow administration portlets](#)
- ◆ [Auditing runtime processes](#)
- ◆ [Generating runtime exception reports](#)

Using the workflow administration portlets

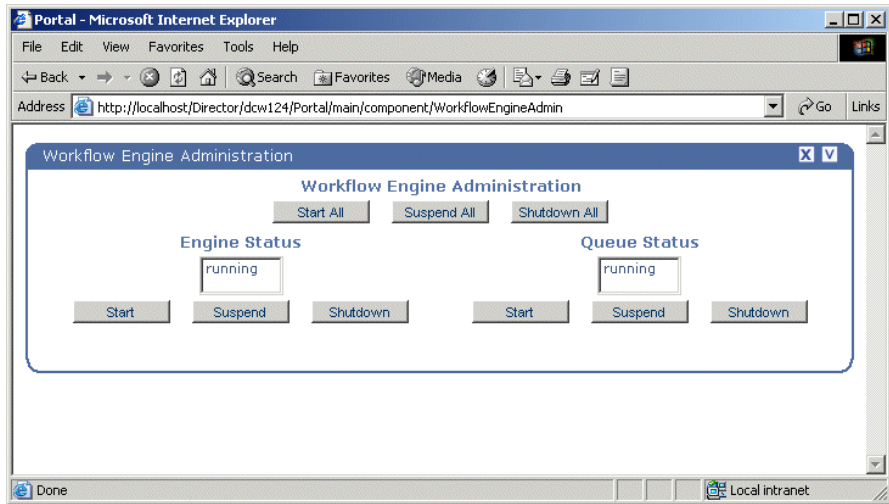
The exteNd Director install includes two workflow administration portlets for managing the engine, queue, and runtime processes:

- ◆ Engine and Queue Administration Console (WorkflowEngineAdmin)
- ◆ Workflow Administration Client (WorkflowAdminClient)

Workflow administrators can access these portlets from the Director Administration Console (DAC).

 For more information, see the chapter [about the Director Administration Console](#) in *Developing exteNd Director Applications*.

Engine and Queue Administration Console



The engine and queue administration console provides the following functionality:

- ◆ Start, suspend, or shut down the workflow queue
- ◆ Start, suspend, or shut down the workflow engine
- ◆ Start, suspend, or shut down both

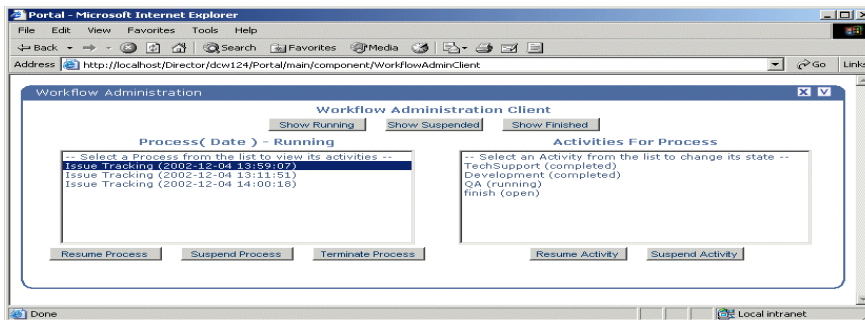
Start functions Start the selected operation. This option starts an operation from scratch, resumes a suspended operation, or restarts an operation that was shut down.

Suspend functions Suspend the specified operation. Any messages sent to the engine or queue are stored but not executed until you select **Start**.

Shutdown functions Shut down the specified operation. Any messages sent to the engine or queue during the shutdown phase are lost.

NOTE: You cannot shut down the engine or the queue if it is in the suspended state. In other words, the engine or queue must be running in order for the shutdown function to work.

Workflow Administration Client



This portlet allows you to manipulate the execution of a process instance by allowing you to:

- ◆ See lists of running, suspended, and finished process instances
- ◆ See a list of the activities and their states in a selected process instance
- ◆ Suspend or resume a process instance
- ◆ Suspend or resume an activity in a process instance
- ◆ Terminate a process instance

Process functions

Process functions operate on all workitems associated with the process, unless that work is already displayed in a user's queue. In this case, workitems are not affected by suspending or resuming either processes or activities. Essentially, the user owns this work, and no existing functions (apart from locking) can prevent the user from updating and forwarding this work.

Suspending a process Suspending a process tells the Workflow subsystem that no updates should be made to workitems executing that process. Resuming a process returns it to the open state. The effect of suspending and resuming is apparent only when a user forwards work from one activity to another.

For example, assume there are two work queues, workqueue-A and workqueue-B. Also assume that workitem-1 is displayed in workqueue-A, and a forward on workitem-1 would normally result in the work appearing in workqueue-B.

If the workflow process currently being executed by workitem-1 is suspended (by someone selecting the Suspend Process button on the Process Administration Console), a forward on workitem-1 would result in the work disappearing from workqueue-A but not appearing in workqueue-B.

Resuming a process Resuming a process returns it to its original state. Continuing with the same example, assume the workflow process is resumed (by selecting the Resume Process button on the Process Administration Console). A refresh of workqueue-B will reveal workitem-1, ready for updating and forwarding by the user.

Terminating a process Terminating a process tells the Workflow subsystem to remove the process instance and its associated activities from the engine queue process. When a process is terminated, it cannot be recovered.

Activity functions

Suspending and resuming activities is similar to suspending and resuming processes, but enables more precision. An activity can be in one of the following states:

This state	Means the workitem
Open	Is not yet running and has not been suspended
Completed	Has been completed and forwarded to the next activity; cannot be suspended
Running	Is active and cannot be suspended
Suspended	Has been suspended; resuming the activity will reopen the workitem

Suspending an activity Suspending an activity tells the Workflow subsystem not to forward work to this activity in this process. Only open activities may be suspended; attempting to suspend a running or completed activity will have no effect. Other combinations may also be ignored (such as attempting to suspend a process or activity that is already suspended).

IMPORTANT: Java activities cannot be suspended, so you cannot access this type of activity using the workflow administration client portlet or the administration APIs.

Continuing with the example from the preceding section, assume there is an additional work queue (workqueue-C) but that workitem-1 is currently displayed in workqueue-A as before. Further assume that a forward from workqueue-B would result in the work appearing in workqueue-C (giving a simple A-B-C queue sequence).

If the workflow activity for workqueue-C is suspended (by someone selecting the Suspend Activity button), then a forward on workitem-1 (currently in workqueue-A) would result in the work showing up in workqueue-B, as expected. However, a forward from workqueue-B would not result in workitem-1 showing up in workqueue-C, since the associated activity is currently suspended.

Resuming an activity Now assume that the activity for workqueue-C is resumed (by someone selecting the Resume Activity button). A refresh of workqueue-C will then reveal workitem-1, ready for updating and forwarding by the user.

Auditing runtime processes

The Workflow API provides the `EbiAuditDelegate.getAuditInfo()` method for getting audit information about workflow processes, activities, workitems, and workitem properties. Here is the method declaration:

```
public Document getAuditInfo(  
    String processId, String runNumber, String activityName, Date  
    startDate, Date endDate)  
    throws com.sssw.wf.client.EboAuditLoggerException
```

The DTD for the return document is available in your exteNd Director project directory at `library/WorkflowService/DTD/workflow-auditlist.dtd`.

NOTE: All parameter values are optional. To return all information, specify `null` for the parameter.

Generating runtime exception reports

The Workflow API includes exception-handling classes for the Workflow subsystem portlets, including `EboEngineException`, `EboQueueException`, and `EboWorkitemException`. By default, exception messages are written out to the server console. But you can have the messages sent via e-mail to a specified user. The setting for this option is in `config.xml`.

➤ **To configure exception messages to be sent to an administrator:**

- 1 Outside exteNd Director, open the following file:

```
My_Project/library/WorkflowService/WorkflowService-  
conf/config.xml
```

- 2 Uncomment the following lines:

```
<!--  
  <property>  
    <key>WorkflowService/administrator-host</key>  
    <value>SomeHost</value>  
  </property>  
  <property>  
    <key>WorkflowService/administrator-address</key>  
    <value>SomeAddress</value>  
  </property>  
-->
```

NOTE: After you uncomment the properties, you can access them in the XML Editor in exteNd Director.

- 3 Set the two properties to a valid user or admin group with a valid e-mail address. For example:

```
<property>  
  <key>WorkflowService/administrator-host</key>  
  <value>SMTP mail server hostname</value>  
</property>  
<property>  
  <key>WorkflowService/administrator-address</key>  
  <value>admin@addresss.com</value>  
</property>
```

- 4 Save the file.

||| Applications

- Chapter 8, “Content Life Cycle Application”

8

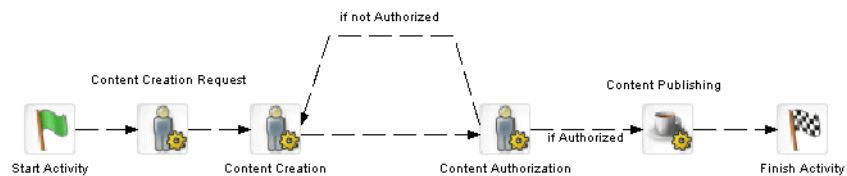
Content Life Cycle Application

This chapter describes how to run the installed Content Life Cycle workflow application and how to access the application sources. It has these sections:

- ◆ [About the Content Life Cycle application](#)
- ◆ [Running the application](#)
- ◆ [Content life Cycle application sources](#)

About the Content Life Cycle application

Content Life Cycle is a workflow portlet application that allows different users to complete activities representing the life cycle of a document. Here is what the process looks like in the Workflow Modeler:



Sample Content Management Process

Each of the User activities is associated with a different user. Let's assume the following users:

- ◆ Content Creation Request: sample
- ◆ Content Creation: sample2

- ◆ Content Authorization: sample3

Here's how the application works:

- 1 The user **sample** creates a new process and completes a form that requests the creation of a document. Sample then forwards the workitem.
- 2 Sample2 creates a document and adds it to the Content Management subsystem. Sample2 then forwards the workitem.
- 3 Sample3 reviews the document and indicates whether the document is authorized for publication, and forwards the workitem to the next activity:

The two links from the Content Authorization activity use the **Flow** scoped path to test the value of a workitem document property called *authorized*. If the value is true the link to the Content Publishing Java activity is followed. If it is false the link back to the Content Creation activity is followed.

- 4 When the content is authorized, the Content Publishing activity (a Java activity) publishes the document using the CM subsystem APIs.

The Java activity uses a scoped path to point to the executable Java class in the project resource set.

Running the application

Before running the Content Life Cycle application, you need add some sample users to your server directory realm.

➤ To add sample users:

- 1 Use your application server tools or the Director Administration Console (DAC) to add the following sample users:
 - ◆ sample

This is the user specified in the workflow process descriptor. The other two can have any name you want, for example:

 - ◆ sample2
 - ◆ sample3

➤ To run the Content Life Cycle application:

- 1 In three separate browsers log in to exteNd Director MyPortal using each sample user.
- 2 For the **sample** user create a personal page called **workflow** and add these installed portlets to the content:
 - ◆ **WorkflowStartProcess**
 - ◆ **WorkflowQueue**

- 3 Repeat steps 1 and 2 for the other two users, but just add the WorkflowQueue portlet to the personal pages.
- 4 Run the portal personal page in each browser. In the **sample** user browser go to the Workflow Start Process, select the process from the dropdown list, and click **Start**.

This creates a workitem in the WorkflowQueue portlet.

NOTE: You can create as many instances as you want.

- 5 Select the workitem and click **View Item**. This displays a workitem form for assigning a document.
- 6 Complete the form, specify one of your workflow users as an addressee, and choose **Forward**.
- 7 Follow the process through content creation, approval, and publish:
 - ◆ Click **Refresh** in the WorkflowQueue portlet in the appropriate browser.
 - ◆ Select the workitem and click **View Item**.
 - ◆ Complete each form, specify an addressee, and forward where appropriate.

Dynamic addressing of workitems

The Content Life Cycle application allows users to specify the addressee for the next activity at runtime. In the process definition, the **addressee** field for the User activity uses the **Flow** scoped path to scope to a workflow property called *target_addressee*.

The value of this property is set by the portlet source for the User activity. Whatever the value is at the time the activity gets forwarded determines the addressee. When the portlet sets a new addressee, it sets a workitem property on the EbiWorkitemDelegate:

```
EbiProperty addrProp = new EboProperty( WORKITEM_ADDRESSEE,
    m_addressee, EboConstants.ATT_STRING, false );
delegate.setProperty( addrProp, WFcontext );
```

TIP: You can also use the Set Workitem Value action in the Rules subsystem to dynamically set workitem properties. For more information, see the chapter on [installed actions](#) in the *Rules Guide*.

Content life Cycle application sources

The application source files and artifacts are contained in JAR files at:

```
install_dir/Director/templates/Director/TemplateResources/
```

Here is a summary:

Parent directory and JAR	Contents
workflow-portlets/workflow_portlets.jar	<ul style="list-style-type: none">◆ Class files and XSL descriptors for the WorkflowStartProcessPortlet and WorkflowQueuePortlet.◆ WorkflowUsers security role descriptor
workflow-portlets/workflow_portlets_src.jar	<ul style="list-style-type: none">◆ Java sources for WorkflowStartProcessPortlet and WorkflowQueuePortlet.
workflow-sample-portlets/workflow_sample_portlets.jar	<ul style="list-style-type: none">◆ Class files for the User activities◆ activitypolicy.xml for specifying the client classes for the User activities◆ The Content Life Cycle process descriptor
workflow-sample-portlets/workflow_sample_portlets_src.jar	<ul style="list-style-type: none">◆ Java sources for the User activity classes


IV Reference

- [Chapter 9, “Workflow Tag Library”](#)

9

Workflow Tag Library

This chapter describes how to use the JSP tags contained in `WorkflowTag.jar`.

 For background information, see the chapter on [using the exteNd Director tag libraries](#) in *Developing exteNd Director Applications*.

These are the JSP tags:

- ◆ `addressee`
- ◆ `createProperty`
- ◆ `forwardWorkitem`
- ◆ `getDocument`
- ◆ `getProcessList`
- ◆ `getProperty`
- ◆ `getPropertyList`
- ◆ `getQueueStatus`
- ◆ `getWorkitem`
- ◆ `getWorklist`
- ◆ `hasDocument`
- ◆ `hasProperty`
- ◆ `isDocumentLocked`
- ◆ `isWorkitemLocked`
- ◆ `setProperty`
- ◆ `startProcess`
- ◆ `updateDocument`
- ◆ `workitemLock`

addressee

Description Specifies a workitem addressee that is passed to the tag within which it is nested. This tag is a nested tag. It can only be used as the child of another tag.

The parent tag must have a method of signature `addAddressee(String x)`. The addressee tag calls that method on the parent tag and passes in the addressee.

See also [getWorklist](#)

Syntax `<prefix:addressee>addressee</>`

Example This example shows how to specify addressee nested tags in the `getWorklist` parent tag:

```
<epwf:getWorkList id="example" iterate="true" >

<epwf:addressee>user1</epwf:addressee>
<epwf:addressee>user2</epwf:addressee>

  Activity = <%=activity%><br/>
  workitemid = <%=workitemid%><br/>
  name = <%=name%><br/>
  lockedby = <%=lockedby%><br/>
  message = <%=message%><br/>
</epwf:getWorkList>
```

createProperty

Description Creates an `EbiProperty` object for a workitem or document using the specified values for property name, value, and type.

If `docName` is provided, the property object is created for the document. The `EbiProperty` object is returned in a default scripting variable called **newProperty**, or the name provided in the `id` attribute.

This tag wraps a version of `createProperty()` on the `EbiWorkitemDelegate` interface.

Syntax `<prefix:createProperty workitemID="workitemID" docName="docName" propName="propName" value="value" type="type" immutable="immutable" id="id"/>`

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
docName	No	Yes	Name of the document to which to add the property
propName	Yes	Yes	Name for the property
value	Yes	Yes	Property value
type	Yes	Yes	Data type for the property value; for valid entries, see EbiProperty in the API documentation
immutable	No	Yes	Boolean value (true or false) that specifies whether the property is immutable; default is false
id	No	No	Name of the variable in which to store the returned property object; default name is newProperty

Example

This example creates a workitem property and gets the property name from the returned object:

```
<epwf:createProperty id="example"
workitemID="<%=wid.getWorkitem().getId()%>" propName="wiproperty2"
value="testvalue" type="string" / >

<% com.sssw.wf.api.EbiProperty prop =
(com.sssw.wf.api.EbiProperty)pageContext.getAttribute("example");%>
<br/>
<%=prop.getPropertyName()%>
```

forwardWorkitem

Description Forwards a workitem to the next activity and returns a boolean representing whether the attempt to forward was successful.

The value is returned on a scripting variable called **wi_forward** or the name provided in the `id` attribute.

This tag wraps the `forward()` method on the `EbiQueueDelegate` interface.

Syntax

```
<prefix:forwardWorkitem workitemID="workitemID" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Current workitem ID
id	No	No	Name of the variable in which to store the return value; default name is wi_forward

Example

```
<epwf:forwardWorkitem id="forward" workitemID='<%=x%>' />  
<%=pageContext.getAttribute("forward") %>
```

getDocument

Description Returns a document for a specified workitem ID. By default it returns the document as an object of type `org.w3c.dom.Document`. If `returnDOM` is set to `false`, the document is returned as a `String`. The value is returned on a scripting variable called **wi_document** or the name provided in the `id` attribute.

This tag wraps `getDocument()` on the `EbiWorkitemDelegate` interface.

Syntax

```
<prefix:getDocument workitemID="workitemID" docName="docName"  
returnDOM="returnDOM" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
docName	Yes	Yes	Name of the document
returnDOM	No	Yes	Boolean that returns the document as an object of type <code>org.w3c.dom.Document</code> (true) or as a String (false); the default is false
id	No	No	Name of the variable in which to store the return value; default name is wi_document

Example

This example gets a workitem document as a String:

```
<epwf:getDocument id="example"
workitemID="<%=wid.getWorkitem().getId()%" docName="mydoc1"
returnDOM="false" />
<%=pageContext.getAttribute("example")%>
```

getProcessList

Description

Retrieves a list of available workflow processes. Returns the list as a DOM or as variables that can be set within the tag. The result is returned using a scripting variable called **processList** or by a value set in the `id` attribute.

Tag options

Do one of the following:

- ◆ Set **returnDOM** to true (returns DOM) or false (returns String XML). In this case the `iterate` attribute must be false.
- OR**
- ◆ Set **iterate** to true to return results via variables in the body tag. In this case `returnDOM` is not used.

This tag wraps the `getProcessDefinitions()` method on the `EbiWorkflowEngineDelegate` interface.

Syntax

```
<prefix:getProcessList returnDOM="returnDOM" iterate="iterate"
id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
returnDOM	No	Yes	<p>Boolean value (true or false) that indicates whether to return the list as a DOM (true) or as an XML String (false)</p> <p>The DOM adheres to workflow-process_4_0.dtd</p> <p>NOTE: If returnDOM is set to true, the iterate attribute must be set to false</p>
iterate	No	Yes	<p>Boolean value (true or false) that indicates whether this tag will operate as a body tag so that each row can be processed separately; default value is false</p> <p>If the iterate attribute is set to true, the following values can be accessed from within the getProcessList tag:</p> <ul style="list-style-type: none"> ◆ processid ◆ name <p>Each of these variables has a scope of NESTED</p> <p>NOTE: When the iterate attribute is set to true, returnDOM is not used</p>
id	No	No	<p>Name of the variable in which to store the returned boolean value; the default name is processList</p>

Example

This example gets a process list using iterated nested variables:

```
<epwf:getProcessList iterate="true" >
  ProcessID = <%=processid%><br/>
  Name = <%=name%><br/>
</epwf:getProcessList>
```


getProperty

Description Gets a property object of type EbiProperty for a workitem or (if docName is provided) for a document. The value is returned on a scripting variable called **property** or the name provided in the id attribute.

This tag wraps the getProperty() method on the EbiWorkitemDelegate interface.

Syntax

```
<prefix:getProperty workitemID="workitemID" propName="propName"  
docName="docName" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
propName	Yes	Yes	Name of the property
docName	No	Yes	If this is a document property, the name of the document associated with the property
id	No	No	Name of the variable in which to store the returned property object; the default name is property

Example

This example gets a workitem property name and value from the returned property object:

```
<epwf:getProperty id="example"  
workitemID="<%=wid.getWorkitem().getId()%>" propName="wiproperty"  
>  
  
<%= ((com.sssw.wf.api.EbiProperty)pageContext.getAttribute("example"  
) ).getPropertyName()%> = name  
<%= ((com.sssw.wf.api.EbiProperty)pageContext.getAttribute("example"  
) ).getPropertyValue()%> = value
```

getPropertyList

Description

Returns a List (java.util.List) of EbiProperty objects for a given workitem and/or related document. Either the forWorkitem or forDocuments attribute must be set to true. If both are set to true, all properties will be returned for the workitem and all related documents. List is returned on a scripting variable called **propertyList** or the name provided in the id attribute.

This tag wraps the getAllProperties(), getWorkitemProperties, and getDocumentProperties on the EbiWorkitemDelegate interface, depending on the attributes provided.

Syntax

```
<prefix:getPropertyList workitemID="workitemID"  
forWorkitem="forWorkitem" forDocuments="forDocuments" docName="docName"  
iterate="iterate" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
for Workitem	No	Yes	Boolean value that specifies workitem properties; if this attribute and the forDocuments attribute are true and no docName is provided, all workitem properties are returned
for Documents	No	Yes	Boolean value that specifies document properties; if this attribute is true, you can also set the docName attribute
docName	No	Yes	Name of the document whose properties you want returned (only the properties for the specified document are returned); if you do not specify a docName, all properties are returned

Attribute	Required?	Request-time expression values supported?	Description
iterate	No	Yes	<p>Boolean value (true or false) that indicates whether this tag will operate as a body tag so that each row can be processed separately</p> <p>The default is false, which returns a List object on the scripting variable named propertyList or the name provided in the id attribute; List contains objects of type EbiProperty</p> <p>If iterate is set to true, the following values will be available in the body tag:</p> <ul style="list-style-type: none"> ◆ name: the property name ◆ type: the property type ◆ value: the property value ◆ docName: the name of the document to which the property is related <p>Each of these variables has a scope of NESTED</p>
id	No	No	<p>Name of the variable in which to store the List; the default name is propertyList</p>

Examples

Property list using iterated nested variables:

```
<epwf:getPropertyList id="example1" iterate="true"
workitemID="<%=wid.getWorkitem().getId()%>" forWorkitem="true"
forDocuments="true" >
  name = <%=name%><br/>
  value = <%=value%><br/>
  docName = <%=docName%><br/>
</epwf:getPropertyList>
```

Property list as a Java List:

```
<epwf:getPropertyList id="example2" iterate="false"
workitemID="<%=wid.getWorkitem().getId()%>" forWorkitem="true"
forDocuments="true" />
<%=example2.size()%> = size
```

getQueueStatus

Description Checks the status of the workflow engine. Returns the word **running**, **shutdown**, or **suspended**. The value is returned on a scripting variable called **queueStatus** or the name provided in the id attribute.

This tag wraps getQueueStatus on the EbiQueueDelgate interface.

Syntax

```
<prefix:getQueueStatus id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
id	No	No	Name of the variable in which to store the return value; the default name is queueStatus

Example

```
<epwf:getQueueStatus id="queue" />  
<%=pageContext.getAttribute("queue") %>
```

getWorkitem

Description Returns an object of type EbiWorkitemDelegate for the specified workitemID. The value is returned on a scripting variable called **workitem** or the name provided in the id attribute.

This tag wraps getWorkitem() on the EbiQueueDelgate interface.

Syntax

```
<prefix:getWorkitem workitemID="workitemID" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
id	No	No	Name of the variable in which to store the return value; the default name is workitem

Example

```
<epwf:getWorkitem id="get" workitemID='<%=x%>' />  
<%=pageContext.getAttribute("get") %>
```

getWorklist

Description Retrieves a list of workitems from a specified queue. The result is returned using a default scripting variable called **workList** or by a value set in the id attribute.

You can embed the workitem addressee in this tag using the **addressee** tag.

This tag wraps a version of getWorklist() on the EbiQueueDelegate interface. The method version used depends on which tag attributes are provided.

Tag options Two steps are required:

Step	What to do
1	First make sure one of the following is true: <ul style="list-style-type: none">◆ The user is logged in so that the workitem list can be retrieved from the workflow context. OR◆ The activity attribute is set. OR◆ The addressee or name attribute is set. The ignoreSuspended attribute is used in this instance, with default value true. To pass the addressee you can nest the addressee tag inside the getWorklist tag.
2	Then do one of the following : <ul style="list-style-type: none">◆ Set the returnDOM attribute to true (return as XML DOM) or false (return as XML string). When this attribute is set, the iterate attribute must be set to false. OR◆ Set iterate to true. The results are returned via variables in the body tag. In this case, returnDOM is not used. OR◆ Set returnList to true. In this case a list of EbiWorkitemDelegate objects is returned in an object of type java.util.List.

The getWorklist tag wraps a version of getWorklist() on the EbiQueueDelegate interface. The method version used depends on which tag attributes are provided.

Syntax

```
<prefix:getWorklist activity="activity" name="name"  
returnDOM="returnDOM" returnList="returnList" iterate="iterate"  
priority="priority" ignoreSuspended="ignoreSuspended" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
activity	No	Yes	Name of the activity; this corresponds to the Name property for the activity defined in the Workflow Modeler
name	No	Yes	Name of the workitem; this corresponds to the WorkName property defined for the process in the Workflow Modeler
returnDOM	No	Yes	<p>Boolean value (true or false) that indicates whether or not to return the list as a DOM (true) or as an XML String (false)</p> <p>The DOM adheres to workflow-worklist_4_0.dtd</p> <p>If returnDOM is set to true, the iterate attribute must be set to false</p>
returnList	No	Yes	A boolean value (true or false) that when set to true returns a list of EbiWorkitemDelegate objects as an object of type java.util.List
iterate	No	Yes	<p>A boolean value (true or false) that indicates whether this tag will operate as a body tag so that each row can be processed separately; the default value is false</p> <p>If the iterate attribute is set to true, the following values can be accessed from within the getWorkList tag:</p> <ul style="list-style-type: none"> ◆ activity ◆ name ◆ workitemid ◆ message ◆ lockedby <p>Each of these variables has a scope of NESTED</p> <p>NOTE: When the iterate attribute is set to true, returnDOM is not used</p>

Attribute	Required?	Request-time expression values supported?	Description
priority	No	Yes	Provided to support user-implemented priority schemes; not supported in the current version
ignore Suspended	No	Yes	Boolean value (true or false) that indicates whether to ignore suspended activities in the returned list
id	No	No	Name of the variable in which to store the result; the default name is workList

Examples

Worklist using iterated nested variables:

```
<epwf:getWorkList iterate="true" >
  Activity = <%=activity%><br/>
  Name = <%=name%><br/>
  Workitemid = <%=workitemid%><br/>
  <#! String x;%>
  <% x = (String) pageContext.getAttribute("workitemid"); %>
  Message = <%=message%><br/>
  Lockedby = <%=lockedby%><br/>
</epwf:getWorkList>
```

Worklist as a DOM:

```
<epwf:getWorkList id="list" iterate="false" returnDOM="true" />
// access the DOM
xml string = <%org.w3c.dom.Document doc = (org.w3c.dom.Document)
pageContext.getAttribute("list");%>
<%=doc.getFirstChild().getNodeName()%>
```

Worklist as a Java List object:

```
<epwf:getWorkList id="list2" returnList="true" />
<% for (int x=0;x<list2.size();x++){
  // parse the List
  com.sssw.wf.client.EbiWorkitemDelegate wid =
  (com.sssw.wf.client.EbiWorkitemDelegate) list2.get(x); %>
  Wid <%=x%> activity name = <%=wid.getActivityName()%>
<% } %>
```

Worklist as an XML string:

```
<epwf:getWorkList id="list3" iterate="false" returnDOM="false" />
xml string = <%= (String) pageContext.getAttribute("list3") %>
```

Worklist using an embedded **addressee** tag:


```

<epwf:getWorkList iterate="true" >
<epwf:addressee>sample</epwf:addressee>
  Activity = <%=activity%><br/>
  workitemid = <%=workitemid%><br/>
  name = <%=name%><br/>
  lockedby = <%=lockedby%><br/>
  message = <%=message%><br/>
</epwf:getWorkList>

```

hasDocument

Description

Checks whether a document exists for workitem and returns a boolean. The value is returned on a scripting variable called **hasDocument** or the name provided in the id attribute.

This tag wraps hasDocument() on the EbiWorkitemDelegate interface.

Syntax

```

<prefix:hasDocument workitemID="workitemID" docName="docName"
returnDOM="returnDOM" id="id"/>

```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
docName	Yes	Yes	Name of the document
returnDOM	No	Yes	Boolean that returns the document as an object of type org.w3c.dom.Document (true) or as a String (false); default is false
id	No	No	Name of the variable in which to store the return value; the default name is hasDocument

Example

```

<epwf:hasDocument workitemID="<%=wid.getWorkitem().getId()%>"
docName="mydocument" />
<%=pageContext.getAttribute("hasDocument")%>

```

hasProperty

Description

Checks whether a property exists for a workitem or (if docName is provided) for a document. Returns a boolean. The value is returned on a scripting variable called **hasProperty** or the name provided in the id attribute.

This tag wraps hasProperty() and hasDocument Property on the EbiWorkitemDelegate interface.

Syntax

```
<prefix:hasProperty workitemID="workitemID" docName="docName"  
propName="propName" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
docName	No	Yes	Name of the document whose property you want to check; use only for document properties
propName	Yes	Yes	Name of the property
id	No	No	Name of the variable in which to store the return value; the default name is hasProperty

Example

```
<epwf:hasProperty id="status"  
workitemID="<%=wid.getWorkitem().getId()%">" propName="wiproperty2" />  
<%=pageContext.getAttribute("status")%>
```

isDocumentLocked

Description

Checks whether a workitem document is locked—and if the `userID` attribute is set, also checks whether the document is locked by a particular user. Returns a boolean. The value is returned on a scripting variable called `doc_islocked` or the name provided in the `id` attribute.

This tag wraps `isDocumentLocked()` or `isDocumentLockedBy()` on the `EbiWorkitemDelegate` interface.

Syntax

```
<prefix:isDocumentLocked workitemID="workitemID" docName="docName"
userID="userID" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
docName	Yes	Yes	Name of the document
userID	No	Yes	UserID; if not set, the tag checks the document status only
id	No	No	Name of the variable in which to store the return value; the default name is <code>doc_isLocked</code>

Example

```
<epwf:isDocumentLocked id="doclocked" workitemID='<%=x%>'
docName="doc1" />
<%=pageContext.getAttribute("doclocked") %> <br/>
```

isWorkitemLocked

Description Checks whether a workitem is locked—and if the userID attribute is set, also checks whether the workitem is locked by a particular user. Returns a boolean. The value is returned on a scripting variable called **wi_islocked** or the name provided in the id attribute.

This tag wraps isLocked() or isLockedBy() on the EbiWorkitemDelegate interface.

Syntax

```
<prefix:isWorkitemLocked workitemID="workitemID" userID ="userID" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
userID	No	Yes	UserID to check; if not set, the tag checks the workitem status only
id	No	No	Name of the variable in which to store the return value; the default name is wi_isLocked

Example

```
<epwf:isWorkitemLocked id="worklock" workitemID='<%=x%>' />  
<%=pageContext.getAttribute("worklock") %> <br/>
```

setProperty

Description Stores an existing EbiProperty object for a workitem (or for a document if docName is provided) using the specified EbiProperty object. Returns a boolean representing success or failure. Return value is stored in a scripting variable called **setProperty** or the name provided in the id attribute.

If the userID attribute is not specified, the current user is used.

This tag wraps setProperty() and setDocumentProperty() on the EbiWorkitemDelegate interface.

Syntax

```
<prefix:setProperty workitemID="workitemID" property="property" forDocument="forDocument" userID ="userID" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
property	Yes	Yes	Name of the property to add
for Document	No	Yes	Boolean value that specifies whether the property is a document property; the default is false NOTE: The property must have been previously associated with the specified document; see "createProperty" on page 100
userID	No	Yes	UserID If set, the tag sets the document or workitem property for the user; otherwise, it uses the current context
id	No	No	Name of the variable in which to store the return value; the default name is setProperty

Example

This example sets a property from an attribute provided on the page:

```
<epwf:setProperty id="set"
workitemID="<%=wid.getWorkitem().getId()%"
property="<%= (com.sssw.wf.api.EbiProperty)pageContext.getAttribute(
"myatt")%" / >
<%=pageContext.getAttribute("set")%">
```

startProcess

Description

Starts a workflow process with a specified processID. Starts the process with a document if the docName attribute is specified, as well as one of the following attributes representing the document:

- ◆ document of type org.w3c.dom.Document
- ◆ docstring of type String
- ◆ url to the document

Returns an int representing success or failure:

Value	Meaning
0	Success
-1	Engine not running; unable to start process
-2	Process definition not found
-3	Process suspended

The value is returned on a scripting variable called **processStarted** or the name provided in the id attribute.

This tag wraps startProcess() on the EbiWorkflowEngineDelegate interface.

Syntax

```
<prefix:startProcess processID="processID" docName="docName"  
document="document" docstring="docstring" url ="url" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
processID	Yes	Yes	Workitem ID
docName	No	Yes	Name of the starting document
document	No	Yes	Starting document, as a DOM
docstring	No	Yes	Starting document, as a String
url	No	Yes	URL to the starting document (DOM)
id	No	No	Name of the variable in which to store the return int value; the default name is processStarted

Example

```
<epwf:startProcess id ="start"  
processID="c373e9ea737c902a8f7af0aa8c836fd6" />  
<%=pageContext.getAttribute("start") %>
```

updateDocument

Description

Updates or adds a document for the specified workitem ID and document name and returns a boolean representing success. The return value is stored in a scripting variable called **wi_document** or the id you specify.

To add a new document, set the **addMode** attribute to true. To update an existing document, set it to false (default).

The document may be specified using one of these attributes:

- ◆ document as an XML string or a DOM object
- ◆ String identifier
- ◆ url of type java.net.URL

If the **userID** attribute is not specified, the current user is used.

This tag wraps `addDocument()` and `updateDocument()` on the `EbiWorkitemDelegate` interface.

Syntax

```
<prefix:updateDocument workitemID="workitemID" docName="docName"  
document="document" identifier="identifier" url="url" userID="userID"  
addMode="addMode" id="id" />
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
docName	Yes	Yes	Name of the document
document	No	Yes	The actual document; the document can be specified as an XML String or an object of type org.w3c.dom
identifier	No	Yes	The string identifier for information in the document
url	No	Yes	The URL for the document, specified as an object of type java.net.URL

Attribute	Required?	Request-time expression values supported?	Description
userID	No	Yes	The user with whom to associate the document; if not specified, gets the user ID from the context
addMode	No	Yes	Boolean indicating whether to add a new document (true) or update an existing document (false); the default is false.
id	No	No	Name of the variable in which to store the return value; default name is wi_document

Examples

Adding a document with identifier:

```
<epwf:updateDocument id="doc1"
workitemID="<%=wid.getWorkitem().getId()%" docName="identifier"
addMode="true" identifier="myidentifier" / >
<%= pageContext.getAttribute("doc1")%>
```

Updating a document with identifier:

```
// lock the document first
<epwf:workitemLock workitemID="<%=wid.getWorkitem().getId()%"
docName="identifier" documentLock="true" action="lock"/>
// update the document
<epwf:updateDocument id="doc2"
workitemID="<%=wid.getWorkitem().getId()%"
docName="identifiertest3" identifier="newidentifier" / >
<%=pageContext.getAttribute("doc2")%>
```

Adding document with string XML:

```
<epwf:updateDocument id="doc3"
workitemID="<%=wid.getWorkitem().getId()%" docName="string1"
addMode="true" document="<main><node1>this is my
document</node1></main>" / >
<%= pageContext.getAttribute("doc3")%>
```


workitemLock

Description

Locks or unlocks a workitem or the documents associated with a workitem. The action attribute must be set to either **lock** or **unlock**. Returns a boolean value representing success. The value is returned on a scripting variable called **wilocked** or the name provided in the id attribute.

If documentLock is set to true, a docName must be specified.

If the userID attribute is not specified, the object is locked or unlocked using the current user.

This tag wraps lock(), unlock(), lockDocument(), and unlockDocument() on the EbiWorkitemDelegate interface.

Syntax

```
<prefix:workitemLock workitemID="workitemID" action="action"  
workitemLock="workitemLock" documentLock="documentLock"  
docName="docName" userID="userID" id="id"/>
```

Attribute	Required?	Request-time expression values supported?	Description
workitemID	Yes	Yes	Workitem ID
action	Yes	Yes	Specify lock or unlock
workitem Lock	No	Yes	Boolean indicating whether the action is for the workitem (true); the default is false Either this attribute or documentLock (or both) must be set to true
document Lock	No	Yes	Boolean value indicating whether the lock is for a document associated with the workitem (true); the default is false Either this attribute or workitemLock (or both) must be set to true; if documentLock is true, docName must be specified
docName	No	Yes	Name of the document; use if documentLock is true
userID	No	Yes	User to execute the action on (if not specified, the user ID from the context)

Attribute	Required?	Request-time expression values supported?	Description
id	No	No	Name of the variable in which to store the return value; the default name is wilocked

Examples

workitemLock:

```
<epwf:workitemLock id="wilock" workitemID='<%=x%>' action="lock"
workitemLock="true" />
<%=pageContext.getAttribute("wilock") %> <br/>
<% get.addDocument("test", "ID" , "sample"); %>
```

documentLock:

```
<epwf:workitemLock id="doclock" workitemID='<%=x%>' action="lock"
workitemLock="false" documentLock="true" docName="test" />
<%=pageContext.getAttribute("doclock") %>
```

workitemUnlock:

```
<epwf:workitemLock id="wiunlock" workitemID='<%=x%>'
action="unlock" workitemLock="true" />
<%=pageContext.getAttribute("wiunlock") %> <br/>
<!-- get.addDocument("test1456", "ID" , "sample"); -->
```

documentUnlock:

```
<epwf:workitemLock id="docunlock" workitemID='<%=x%>'
action="unlock" workitemLock="false" documentLock="true"
docName="test" />
<%=pageContext.getAttribute("docunlock") %>
```

Index

A

- activity
 - adding in the Workflow Modeler 61
 - administering in workflow 88
 - Composer Service 46
 - Java 43
 - Pageflow 35
 - Rule 42
 - Start 45
 - Synchronize Merge 44
 - User 37
 - Web Service 39
- activity policy
 - API methods and 39
 - in workflow 38
 - XML example 38
- addressee
 - dynamic assigning in workflow 95
 - workflow activity 35
- addressee tag 100
- attached label
 - in Workflow Modeler 72

B

- Bird's Eye View window
 - in Workflow Modeler 79

C

- client, workflow
 - defined 21
- Composer Service activity
 - defining in Workflow Modeler 46
- conditional link
 - in workflow 51
- Content Life Cycle application
 - application sources 96
- createProperty tag 100
- custom tags
 - addressee 100
 - createProperty 100

- forwardWorkitem 102
- getDocument 102
- getProcessList 103
- getProperty 105
- getPropertyList 106
- getQueueStatus 108
- getWorkitem 109
- getWorklist 110
- hasDocument 113
- hasProperty 114
- isDocumentLocked 115
- isWorkitemLocked 116
- setProperty 116
- startProcess 118
- updateDocument 119
- workitemLock 121

E

- EbiContext
 - workflow context defined 23
- EbiQueueDelegate
 - defined 23
- EbiWorkflowEngineDelegate
 - defined 23
- EbiWorkitemDelegate
 - defined 23
- engine
 - workflow 20
- exceptions
 - reporting in workflow 89
- expressions
 - link expressions in workflow 70

F

- floating label
 - in Workflow Modeler 72
- Flow scope
 - lock and unlock properties 30
- forwardWorkitem tag 102
- full layout
 - in Workflow Modeler 75

G

- getDocument tag 102
- getProcessList tag 103
- getPropertyList tag 106
- getProperty tag 105
- getQueueStatus tag 108
- getWorkiList tag 110
- getWorkitem tag 109
- graphical properties
 - in Workflow Modeler 73
- grid feature
 - in Workflow Modeler 78

H

- hasDocument tag 113
- hasProperty tag 114

I

- incremental layout
 - in Workflow Modeler 76
- isDocumentLocked tag 115
- isWorkitemLocked tag 116

J

- Java activity
 - about 81
 - coding 84
 - creating 82
 - defining in Workflow Modeler 43
 - scoped path access 84

L

- labels
 - in Workflow Modeler 71, 72
- layout features
 - in Workflow Modeler 74
- links
 - about 49
 - conditional link 51
 - creating in Workflow Modeler 63
 - expressions 70
 - simple link 50
- locking and unlocking

- in workflow 30

P

- pageflow
 - workflow integration 28
 - Workflow Return activity 29
- Pageflow activity
 - defining in Workflow Modeler 35
- process
 - see workflow process

Q

- queue, workflow
 - see workflow queue

R

- resource view
 - creating in the Workflow Modeler 80
- Rule activity
 - defining in Workflow Modeler 42

S

- scoped path
 - accessing from a Java activity 84
- scoped paths
 - accessing in the Workflow Modeler 69
 - copying in the Workflow Modeler 67
 - Workflow Modeler and 64
- segment
 - creating in Workflow Modeler 64
- setProperty tag 116
- simple link
 - in workflow 50
- Start activity
 - defining in Workflow Modeler 45
- startProcess tag 118
- synchronize merge
 - in workflow 27
- Synchronize Merge activity
 - defining in Workflow Modeler 44
 - in workflow 27

T

tag libraries

Workflow tag library 99

U

updateDocument tag 119

User activity

defining in Workflow Modeler 37

W

Web Service activity

defining in Workflow Modeler 39

workflow

about 15

activities 88

addressing workitems dynamically 95

API summary 23

client 21

client, administering 87

exceptions, reporting 89

getting started 25

pageflow comparison to 21

pageflow integration 28

process 17

process branching 26

queue function

sample application 93

subsystem architecture 20

workitem 21

Workflow Administration Client 87

workflow client

about 21

workflow engine

about 20

shutting down 86

starting in console 86

suspending 86

Workflow Engine and Queue Administration

Console 86

Workflow Modeler 78

Bird's Eye View window 79

Composer service activity 46

graphical properties in 73

Java activity 43

labels 71

layout features 74

Pageflow activity 35

resource view, creating 80

Rule activity 42

scoped paths 64

Start activity 45

Synchronize Merge activity 44

User activity 37

Web Service activity 39

zoom features 78

workflow process 17

administering 87

auditing 89

defining in Workflow Modeler 56

opening in Workflow Modeler 57

property settings in Workflow Modeler 58

saving in Workflow Modeler 57

shutting down 86

validating in the Workflow modeler 71

workflow queue

about 20

process, shutting down 86

process, suspending 86

starting in console 86

Workflow Return activity

using in a pageflow 29

Workflow subsystem

architecture 20

workitem

about 21

defined 21

locking and unlocking in workflow 30

workitemLock tag 121

Z

zoom features

in Workflow Modeler 78

