Novell exteNd Director

5.2

www.novell.com

DEVELOPING EXTEND DIRECTOR APPLICATIONS

Novell®

Legal Notices

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.

www.novell.com

exteNd Director*Developing exteNd Director Applications*
June 2004

Online Documentation: To access the online documemntation 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

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.

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.

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.

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.

Phaos

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

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 Book

Purpose

This book explains how to design and program a Novell® exteNd Director™ application, giving particular emphasis to using the services of the Framework and Portal subsystems.

Audience

This book is for programmers working on exteNd Director applications. It assumes familiarity with Java programming, HTML, XML, and XSL and provides examples using all these technologies.

Introduction

An introduction to Novell exteNd Director that includes conceptual information about the exteNd Director development environment and application architecture

1 About Novell exteNd Director

This chapter provides an overview of Novell exteNd Director. It contains the following sections:

About Novell exteNd Director

Novell exteNd Director is a set of software development tools and programming APIs for building state-of-the-art enterprise applications. exteNd Director provides all of the technologies you need to build Web applications that present a relevant view of business functions to any user on any device. exteNd Director also provides the tools you need to consume **Web Services**, including those created with exteNd Composer.

The exteNd Director application architecture supports many ways of building user views of business functions. For example, by taking advantage of the **user profiling** and **content management** features provided with exteNd Director, you can ensure that individual users see information that is most relevant to their needs. In addition, you can use the **workflow** features of exteNd Director to model business processes, and **rules** to model business decisions. exteNd Director applications can satisfy the hardware requirements of a wide range of users, including conventional desktop users as well as those accessing Web content from wireless devices.

exteNd Director portal

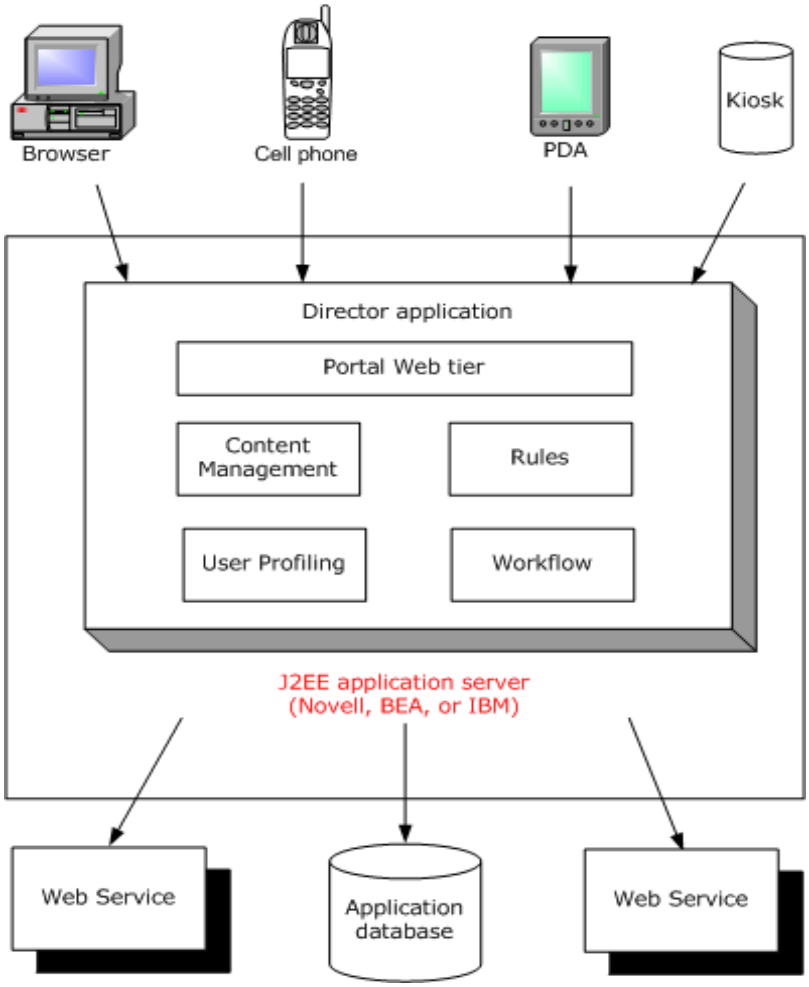
exteNd Director applications typically include a **portal** Web site. The portal is the presentation layer for an exteNd Director application, providing the interface through which users get to the Web content they want to see.

To give you a head start in building applications, exteNd Director provides a default project called **Express Portal**, which is a complete portal application based on the standard exteNd Director project template. You can use this application out of the box to:

- Explore the features of an exteNd Director portal
- Learn how to build a portal application
- Build your own customized production-quality portal by adding and modifying portlets and pages and customizing the presentation layer.

Deployment configurations

The exteNd Director application architecture supports a wide range of deployment configurations, as shown in the following diagram.



Standards compliance

exteNd Director supports several industry standards, including the following:

- J2EE
- Portlets
- XForms

J2EE

exteNd Director is a J2EE-compatible platform for building and deploying enterprise-class applications. The J2EE specification defines the required levels of support that compliant application servers must provide, as well as a programming model for applications that will be deployed to the application server. Applications are deployed into a J2EE-compliant application server as J2EE modules. A J2EE module is a collection of components packaged in an archive. exteNd Director applications are deployed in the following kinds of archives:

- Web archive (WAR)
- Enterprise archive (EAR)

Portlets

exteNd Director is fully compliant with the **Java Portlet 1.0 Specification** which defines the portlet standard.

A portlet is a specialized Java class that processes requests from Web clients and generates dynamic content on a portal page. You can think of portlets as pluggable user interface elements that provide a presentation layer for portal applications. Users can personalize the content and appearance of portlets, based on preferences set by an administrator.

XForms

exteNd Director provides an environment for developing **XForms** 1.0-compliant Web forms. The advantages of the XForms standard include:

- Separate data, logic, and presentation modules
- A powerful event model (so that you don't have to use a lot of scripting for client-side validation or calculations)
- A way to process data in XML formats

exteNd Director subsystems

exteNd Director includes a set of core technologies called *subsystems.* Each exteNd Director subsystem is a logical grouping of a set of software *services.* Each service is a Java class that implements a particular interface defined for exteNd Director applications. The subsystems support a **services-oriented architecture**, which means that you can extend or replace the individual services included with each subsystem. The exteNd Director subsystems can be deployed separately, or can be combined together to form an integrated solution. When you create an exteNd Director project, you select the subsystems your application requires.

The exteNd Director subsystems include several prebuilt user interfaces, as well as comprehensive APIs that let you build J2EE applications from the ground up.

exteNd Director includes the following subsystems:

Subsystem	Description
Content Management	Allows you to create, label, categorize, and display content. Your application can retrieve, display, and update content and display it in various formats based on style sheets. Content is stored in a database associated with your application. You can create simple (standalone) documents, hierarchical documents (such as discussion threads), and compound documents (such as reports with linked attachments). Each document is described by a set of *metadata*—an underlying description or definition. The Content Management subsystem allows you to define custom metadata that organizes documents so that individual users can easily retrieve the content that is most relevant to their needs. 📖 For details on using the Content Management subsystem, see the *Content Management Guide*.
Directory	Provides services for managing user authentication. It works with your application server's security realms to check user IDs and passwords and to add users to the realm. 📖 For details on using the Directory subsystem, see the *User Management Guide*.

Subsystem	Description
Framework	Provides core exteNd Director services. The Framework provides support for caching, system configuration, session management, and other services used by the other subsystems. The Framework is required by all other subsystems. 📖 For details on using the Framework subsystem, see the remaining chapters of this book, as well as the com.sssw.fw package hierarchy in the *API Reference*.
Pageflow	Allows you to graphically model the flow of control for a set of pages that execute within a single portlet. Each page within a pageflow presents a set of controls that allow for user interaction. For example, the pages in a flow might provide a way for the user to display stock quotes or weather forecasts, or access corporate data such as employee information. 📖 For complete details on developing pageflows, see the *Pageflow and Form Guide*.
Portal	Provides Web presentation services. It includes the Portal Aggregator, the Page Manager, the Portal Administrator, and the Portal Personalizer. To take advantage of the services of the Portal subsystem, you typically build your own Web applications. You can configure your Web applications to include the Portal Web tier, which provides end-user portal functionality. 📖 For complete details on developing exteNd Director portal applications, see the *Portal Guide*.
Portlet	Provides development and runtime support for portlets. This support includes the Novell Portlet implementation, as well as the underlying APIs defined by Java Portlet 1.0. When you include the Portlet subsystem, you can create your own custom portlets.
Rule	Allows you to fire business rules. A rule is a conditional formula for making a choice in an exteNd Director application. Rule definitions are stored and edited separately from the portlets that invoke them. Whenever an application decision is in a rule, you can change the logic later, without rewriting and recompiling portlets and pages. Rules consist of *conditions* and *actions*. A condition is a test that determines what action or actions will be taken when the condition is true or false. 📖 For details on using the Rule subsystem, see the *Rules Guide*.
Search	Provides the ability to search content using conceptual pattern matching, a more sophisticated approach than full-text searching, which is traditionally based on keywords. Conceptual searching returns content that is related by meaning and ranked by relevance to the search criteria. The Search subsystem is based on the Autonomy Application Builder toolkit and Dynamic Reasoning Engine (DRE). The Content Management subsystem is integrated with the Search subsystem to provide both SQL-based and conceptual searching capabilities. 📖 For details on using the Search subsystem, see the *Content Search Guide*.

Subsystem	Description
Security	Provides role-based security services to restrict user access to portlets and pages, and ACL-based security services to restrict access to subsystem functionality. You can define security roles and access control lists (ACLs) programmatically or interactively using the Director Administration Console (DAC). 📖 For details on using the Security subsystem, see the *User Management Guide*.
User	Allows you to save information about users in user profiles. When a user has logged in to your exteNd Director application, you can save and update information about the user and the user's usage patterns. User profiles are stored in a database deployed with your application. Typically, user profiles contain two types of information: ◆ **Explicit** Data that users provide, such as an e-mail address or a zip code ◆ **Implicit** Data you collect about their actions, such as how many times they purchase a particular item or view a particular page 📖 For details on using the User subsystem, see the *User Management Guide*.
Workflow	Allows you to graphically model business processes that use simple or rules-based routing (*links*) to move metadata and documents relating to an item of work among workflow *activities*. Participants access their work at activities through workitem queues. A runtime engine manages processes, workitems, and participants. 📖 For details on using the Workflow subsystem, see the *Workflow Guide*.

exteNd Director tools

exteNd Director provides a complete development environment, as well as several prepackaged Web tiers (Web applications) that you can run within a browser.

exteNd Director development environment

At its lowest level, the exteNd Director development environment is a file-system based toolset that includes these utility tools and facilities:

- **Designers and modelers** for building pageflows, portlets, rules, and workflows
- **Graphical and text-based editors** for working on Java files, JSP files, XML files, XSL files, CSS files, WSDL files, HTML files, plain text files, and deployment descriptors
- **Wizards** that help create new files when needed and guide you through complex technologies (such as J2EE)
- **Web Service facilities** for developing, publishing, finding, and running Web Services
- **Project views** that show the structure of a project's source files and the structure of a project's generated archives
- **Project tools** for building projects, generating and validating J2EE archives, and deploying archives to supported J2EE servers
- **Version control integration** that provides access from the exteNd Director development environment to your version control system

exteNd Director Web tiers

exteNd Director includes some prebuilt Web tiers. Each Web tier is a Web application that you can use immediately after deploying your exteNd Director project. Each of the Web tiers runs in a browser.

exteNd Director includes the following Web tiers:

- Express Portal
- Director Administration Console (DAC)
- CMS Administration Console

These tools are intended for use by application developers, system administrators, and content developers respectively.

Internally, these Web tiers use the services of a number of exteNd Director subsystems, including the Framework, Portal, User, Directory, and Content Management subsystems.

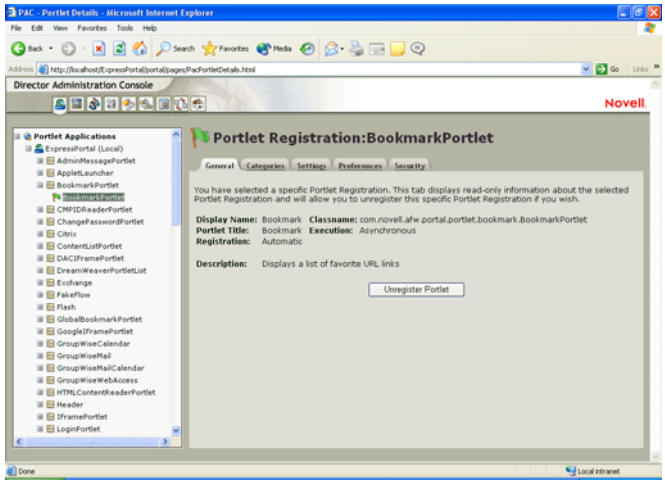
Express Portal

When you install exteNd Director, you get an **Express Portal** application that you can run immediately. You do not need to deploy the application from the exteNd Director development environment to use it. You can simply start your browser and begin using the application right away.

For more information on using the Express Portal, see the chapter on the Express Portal in the *Portal Guide*.

Director Administration Console

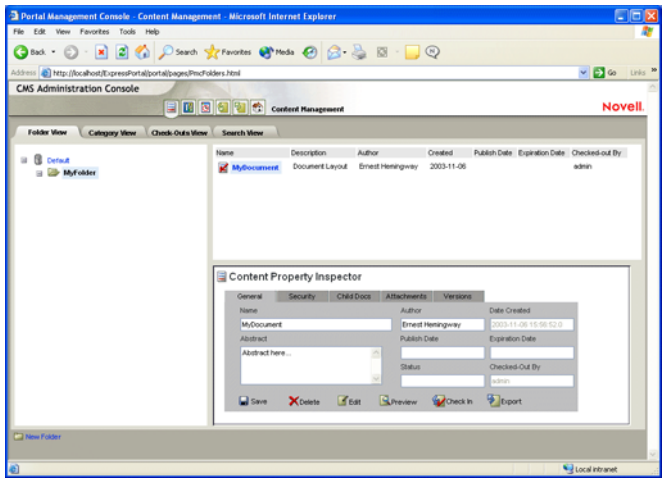
The Director Administration Console (DAC) provides support for administrative tasks such as configuring parameters, managing security access to portal objects, inspecting portlets, pages, and styles, and managing user profiles.



For more information on using the Director Administration Console, see Chapter 21, "About the Director Administration Console".

CMS Administration Console

The CMS Administration Console provides an interface for setting up and maintaining the infrastructure for a content management system. The CMS Administration Console allows you to create, edit, secure, and publish HTML and XML content for your portal applications.



For details on using the CMS Administration Console, see the *Content Management Guide*.

exteNd Director API

To write Java code for exteNd Director applications, you use exteNd Director API classes in your Java code and call their methods. The exteNd Director API provides public classes (and interfaces) organized into several packages, which themselves are organized by subsystem.

The exteNd Director API is based on the Java 2 APIs (J2SE and J2EE). That means it includes classes that inherit from Java 2 classes and implement Java 2 interfaces. If you're familiar with the Java 2 APIs, you'll have a good foundation for understanding and using the exteNd Director API.

📖 For more information on using the exteNd Director API, see Chapter 11, "Coding Java for exteNd Director Applications".

Building an application

An exteNd Director application is packaged in a single **exteNd Director** EAR or WAR file. You can create a new project for the EAR or WAR by running the exteNd Director Project Wizard in the development environment. Alternatively, you can simply open the project for the Express Portal in the development environment and begin working in that project.

If you're new to exteNd Director, you might want to begin by using the Express Portal application right away.

Using the Express Portal application out of the box

You can run the Express Portal application as soon as you finish the installation process. The Express Portal provides a set of Web-based tools you can use to customize the application. For example, you might want to create user pages and shared pages, and then add some predefined portlets to these pages.

Working in the Express Portal project

If you install the exteNd suite using the Express or Custom install, the Express Portal project is added to your suite install directory. You can then open the project in exteNd Director and customize it with your own business logic—for example, by adding new portlets, portal pages, and other Web resources such as JSPs and servlets.

Here are the general steps you need to follow to build your application within the Express Portal project:

1 Install exteNd Director.

📖 For installation instructions, see *Installing Novell exteNd*.

2 Open the Express Portal project in exteNd Director.

📖 To access the Express Portal project in exteNd Director, see the section on opening the Express Portal project in the *Portal Guide*.

3 Add any objects you need for your application within this project. For example, you might want to add pageflows, forms, and portlets to the application.

4 Deploy the application.

NOTE: If you installed the exteNd suite using the **Express** install option, the Express Portal is deployed to the exteNd Application Server at installation time and ready to run. To get started, see the section on starting the Express Portal application in the *Portal Guide*.

📖 For instructions on deployment, see Chapter 20, "Deploying exteNd Director Applications".

Dynamic loading and the resource set exteNd Director provides a special location called the **resource set** that manages application resources you create. The resource set can hold definitions for pageflows, portlets, rules, styles, and other objects required for your application to function properly. The resource set can also hold Java classes that you implement. When you make changes to items in the resource set, you do not need to redeploy the application, since these resources are loaded dynamically.

Creating a new project

If you decide to create a new project, you need to run the exteNd Director Project Wizard. This wizard lets you select the subsystems you want to use and specify configuration properties for these subsystems. After you've finished making your selections, the wizard creates a project that includes the J2EE modules you need for your application.

Once you've run the wizard, you can add additional application-specific J2EE modules to the project. You can also add or remove subsystems or make any other necessary changes to the structure and content of the EAR or WAR, just as you would with any J2EE application.

When you're ready to deploy the application, you can use the deployment tools provided in the exteNd Director development environment to deploy the EAR or WAR to one or more servers.

Here are the general steps you need to follow to create and deploy an **exteNd Director** application in a new project:

1 Install exteNd Director.

   📖 For installation instructions, see the *Installing Novell exteNd*.

2 If you will use the Search subsystem, configure the Autonomy DRE.

   📖 For instructions, see the *Content Search Guide*.

3 Create a new database for your exteNd Director application, and if your JDBC driver expects it, define an ODBC data source name (DSN) for your newly created database. exteNd Director uses a default name of **exteNd Director** for the data source, but you can use your own name.

   This database holds exteNd Director tables that contain data required by several of the subsystems.

   On a Novell exteNd Application Server, this can be different from the deployment database. A typical scenario is to deploy the exteNd Director application to the SilverMaster database, and provide a separate database for the exteNd Director data.

   📖 For instructions on creating the database, see your DBMS documentation.

4 Create a new exteNd Director EAR or WAR project using the exteNd Director Project Wizard with an exteNd Director template.

   📖 For instructions, see "Creating an exteNd Director project" on page 33.

5 Set up dynamic loading so that you can test application resources without frequent redeployments.

   📖 For instructions, see "Dynamic loading of resources and classes" on page 77.

6 Do the required deployment setup for your server. Some setup is done in exteNd Director; for the rest, you use your server's tools.

7 Deploy the application.

   📖 For instructions, see Chapter 20, "Deploying exteNd Director Applications".

II Working with Projects

Explains how to create and update exteNd Director projects

2 Creating exteNd Director Projects

This chapter explains how to create exteNd Director projects. It contains these sections:

About exteNd Director projects

Your work in exteNd Director is organized into *projects*. A project contains:

- Your application objects
- A J2EE archive (used to deploy the application)
- A set of JARs (the JARs included depend on your project configuration choices; for more information see "Subsystem architecture" on page 46)

A project file has an .SPF extension. A single SPF file can contain multiple components and resources of many different types.

You can learn more about projects in general in the chapter on projects and archives in *Utility Tools*.

Default project When you install exteNd Director, a default project called Express.spf is installed on your system and deployed to your server (for some installation types). This project is a complete, working application that you can use as a basis for building your own production portal.

For more information, see the chapter on the using the Express Portal in the *Portal Guide*.

Choosing the project type

exteNd Director provides wizards that build different types of projects. Choosing the right project type includes decisions and knowledge about the:

- Type of application you want to build
- Services your application requires
- Configuration of the application server on which you are deploying

exteNd Director supports the following project types:

Project type	Description	Deployment considerations
Portlet application	Choose this option when you want: ◆ An easy way to distribute your portlets ◆ Your portlets to run **external** to the exteNd Director portal. (If you want your portlets to run local to the portal, choose the Project Wizard described next.) The Portlet Application Wizard creates: ◆ An exteNd Director WAR project. ◆ A J2EE WAR that includes the portlet runtime environment. You can supply an existing WAR, or use the wizard to create a new WAR. 📖 For more information about running portlets externally or locally, see the *Portal Guide*.	Can be deployed in shared, nonshared library, and 3rd party JARs environments. When you run your portlets external to the portal, you can have one of these configurations: ◆ A standalone portlet application WAR. This configuration **requires** a shared library environment with an exteNd Director portal already deployed on the application server. or ◆ A portlet application WAR packaged in an EAR. The EAR also contains the exteNd Director portal WAR. 📖 For more information on setting up a shared library environment, see "Changing a project's shared library configuration" on page 53. For **nonshared library** and **3rd party JARs** environments: ◆ Must be deployed as part of an exteNd Director EAR project.
Project	Choose this option when you want: ◆ To create an EAR or WAR project that includes: ◆ The exteNd Director Portal. ◆ The exteNd Director portlet runtime container. ◆ One or more exteNd Director subsystems. This wizard creates: ◆ An exteNd Director EAR or WAR project. ◆ A customized solution based on your choice of exteNd Director subsystems ◆ 📖 For information on using the Project Wizard, see "Creating projects" on page 31	Supports shared, nonshared library, and 3rd party JARs environments. For **shared library** environments: ◆ One portal application can be deployed on an application server; that means you can have only one of these projects deployed at a time. If this project includes portlets, they will be running local to the portal regardless of whether it is an EAR or WAR project. For **nonshared library** and **3rd party JAR** environments: ◆ You can deploy as many of these projects as you want. ◆ Each application must use its own exteNd Director database—databases cannot be shared. ◆ The resulting EAR or WAR encapsulates all subsystems, application code, and application metadata. 📖 For more information on setting up a shared library or 3rd party JAR environment, see "Changing a project's shared library configuration" on page 53

Creating projects

To create a **portlet application** project, see “Creating a portlet application project” on page 31

To create an **exteNd Director** project, see “Creating an exteNd Director project” on page 33

Using views to what you’re looking for You can use *views* to display personalized lists of items within an exteNd Director project. Views can be used to look at resources in a resource set, or at system configuration and service settings. exteNd Director ships with several predefined views and also allows you to define custom views to display project items that are of particular interest to you.

📖 For information on using views to find items in an exteNd Director project, see Chapter 10, “Working with Views”.

Creating a portlet application project

➢ **To create portlet application project:**

1 In exteNd Director, select **File>New>Project**.

The New Project dialog displays.

2 Select the **Director** tab, choose **Portlet Application**, then click **OK**.

The Project Wizard displays.

If you already have an EAR project open, the wizard asks if you want to add the new project to it. Choosing yes means that the wizard creates the new project within the existing project. Any portlets existing in or added to the portlet application project would run external to the portal.

3 Continue as described in “Selecting the Web App WAR” next.

Selecting the Web App WAR

1 To:

- Choose an existing WAR, click the ellipsis button and navigate to the disk location. Choose the WAR and click **Open**.
- Create a new WAR, click **Create WAR**. Complete the New Project panel and click **Finish**.

  📖 For more information on the New Project panel, see the sections on creating projects and archives in *Utility Tools*.

2 Click **Next** to go to the next wizard panel. See “Specifying the project and archive name” next.

Specifying the project and archive name

➢ **To specify the project and archive name:**

1 On the Project Information panel, specify the following options:

Option	What to specify
Project Name	You can: ◆ Accept the default (the name of the existing Web App WAR from the preceding step)—this will exteNd Director—enable the named portlet WAR. OR ◆ Specify a new name—this will create a new portlet WAR that is exteNd Director–enabled. (It includes a copy of the existing Web App WAR that you specified in the preceding step.) The .SPF extension is automatically appended. This name appears in the source layout. As you enter a project name, the archive name is filled in automatically. You can keep the same name for your archive or enter another one.
Project Location	Specify the directory where you want the project (and other source files) to be located. The wizard creates a project file (with the .SPF extension) in the project location. As you enter a project location, the archive location setting is filled in automatically. You can change these settings. You can do one of the following: ◆ Type the name of the project directory. ◆ Click the ellipsis beside the Project Location text box to select a location. If you specify a project location directory that does not exist, the wizard asks if it should create it. If you do not specify an absolute path, the wizard generates the project in the **exteNd tools\bin** directory.
Archive Name	Specify the name of the archive file that will be generated. You can keep the default archive name (which matches the project name) or enter a new one. The resulting name appears in the archive layout. The .WAR extension is automatically appended to the name.
Archive Location	Enter the location of the project archive or accept the default (the project root directory). The archive location appears in the archive layout of the Navigation Pane after the project has been created.

2 Click **Finish.**

When the wizard completes, the project is open for editing.

What the wizard generates

The wizard generates an exteNd Director–enabled Web application WAR.

At the top level of the WAR you'll see the standard J2EE WEB-INF/lib folder. In the WEB-INF folder you'll see these important files:

File	Description
novell-portlet.xml	An additional, optional portlet deployment descriptor that allows you to specify a broader range of preferences and settings for portlets, such as title bars and preview images
portlet.xml	Required by Java Portlet 1.0
web.xml	Required for all J2EE-compliant WARs

In the conf folder in the WEB-INF folder, you'll see three important exteNd Director files:

File	Description
resourceset.xml	Lets you manage the contents of the resource set. You can add JARs or file extensions that you want to include in the resource set.
config.xml	Sets the configuration properties for the WAR-level services (like the Director Administration Console and CMS Administration Console).
services.xml	Sets properties for the WAR-level services.

 For information about developing portlet applications, see the chapter on developing portal applications in the *Portal Guide*.

Creating an exteNd Director project

➢ **To start the Project Wizard:**

1 Select **File>New>Project**.

2 Select the **Director** tab, then choose **Project** and click **OK**.

3 Continue as described in "Specifying project information" next.

Specifying project information

➢ **To specify project information:**

1 Complete the Project Information panel as follows:

TIP: If you fill in the text boxes in order, subsequent text boxes are filled in automatically with useful default values.

Project setting	What to specify
Project Type	Choose EAR or WAR.
Project Name	Specify the name you want to use for the project file (the .SPF extension is automatically appended). As you enter a project name, the archive name is filled in automatically.

Project setting	What to specify
Project Location	Specify a root directory for the project. The wizard copies files and subdirectories to this location. You can type a path and/or use the Browse button to select a directory location. The location doesn't have to exist. As you enter a project location, the archive location is filled in automatically. If you do not specify an absolute path, the wizard uses **exteNd**'s **tools\bin** directory.
Archive Name	Specify the name of the archive file that will be generated. The .EAR or .WAR extension is automatically added, depending on the project type specified. The Navigation Pane's archive layout displays the archive name. You can keep the default archive name (which matches the project name) or enter a new one.
Archive Location	Specify a directory location for the project archive or accept the default (which is the Project Location).
J2EE Version	If your server supports J2EE 1.3, select J2EE 1.3; otherwise, select J2EE 1.2. **NOTE:** If you want to change the J2EE version of a project at a later time, see the chapter on how to handle J2EE versions in *Utility Tools*.

2 Click **Next**.

If the directories you specified do not exist, the wizard offers to create each of them. Click **Yes** to confirm any new directories.

The next wizard panel displays. See "Specifying the project setup" next.

Specifying the project setup

➢ **To specify the project type:**

1 Complete the panel by choosing either **Typical** or **Custom**:

Setup type	What the wizard does
Typical	Copies the resources for the default set of exteNd Director subsystems and services to your project directory and uses default values for most configuration options. ◆ Content Management (only included with certain product licenses) ◆ Directory ◆ Framework ◆ Pageflow ◆ Portal ◆ Portlet ◆ Rule ◆ Search ◆ Security ◆ User ◆ Workflow (only included with certain product licenses) ◆ exteNd Composer Service
Custom	Lets you choose which subsystem(s) the wizard will copy to your project directory and provides panels for setting their configuration options. If you choose this option, the wizard displays the Subsystem Selection panel: **1** Select the subsystems you'll need for your project, then click **Next**. **2** If you try to eliminate a subsystem that is required by another selected subsystem, you are informed what subsystems depend on the one you're trying to omit. You have to uncheck those dependent subsystems before you can remove any subsystem they require. **NOTE:** For a WAR project, you must select Framework, Directory, Portal, User, and Security.

2 Click **Next** to go to the next wizard panel. See "Specifying application options" next.

Specifying application options

➢ **To specify the application options:**

1 On the Application Options panel, specify the following settings, then click **Next**:

Custom WAR setting	What to specify
Name	The context name for the WAR. This is used in URLs for pages of your application. This name is not editable. For EAR projects, the wizard generates a WAR within the EAR. The WAR's context name is the same as the project name (specified on the Project Information panel) with **Portal** appended to it. For WAR projects, the context name for the WAR is the same as the Project Name (specified on the Project Information panel).
Resources	Select the component collections, tag libraries, and conditions and actions for rules you want to include in your application WAR. The list of available resources depends on the subsystem(s) you've selected. **IMPORTANT:** exteNd Director provides a set of accessory portlets. The exteNd Director Portal uses some of these portlets on its default portal pages. If you want to use these default pages as shipped, be sure to include accessory portlets in the portal application projects you create in exteNd Director. If you do not want to include accessory portlets in exteNd Director projects, your portal administrator should modify default portal pages accordingly.
Template Resources Location	A location for storing available resources that you can add to the project later. By default, this is the TemplateResources subdirectory. You can specify a different location; exteNd Director will copy available resources.

2 Click **Next** to go to the next wizard panel:

- If your project includes the Content Management subsystem, see "Specifying the Content Management Search configuration" on page 36
- Otherwise, see "Directory configuration" on page 38.

Specifying the Content Management Search configuration

The Content Management Search panel has three tabs: the Repository tab, the Synchronization tab, and the Filters tab.

➢ **To specify the Content Management Search configuration:**

1 On the **Repository** tab, you can specify these settings:

Configuration setting	What to specify
Enable link to the search service?	Select **Yes** if you want to use the Autonomy search capabilities with the Content Management subsystem. **IMPORTANT:** If you enable search, make sure you configure the Autonomy DRE to run with your server, as described in the *Content Search Guide*.
Query Engine Host Name	Specify the host name or IP address of the Autonomy DRE (Dynamic Reasoning Engine). The default is **localhost**.
Query Port	Specify the port number on which the DRE expects to receive queries. The default is **52000**.
Index Port	Specify the port number the DRE uses for indexing. The default is **52001**.
Repository Name	Specify the name of the Content Management repository. The value is always **Default**.

2 On the **Synchronization** tab, you can specify these settings:

Configuration setting	What to specify
Synchronization Mode	Specify how changes to documents are made known to the DRE. Values are: ◆ **immediate**—Propagates changes when they occur; recommended when there is a low volume of document additions and updates. ◆ **batch**—Propagates changes to the DRE as a scheduled background task; recommended for environments with a high frequency of changes.
Operations that cause immediate synchronization	When synchronization mode is immediate, select the operations that cause changes to propagate to the DRE. Use the arrow buttons to move operations to and from the Available and Selected lists. For performance reasons, you may not want all operations to be synchronized immediately.
Number of deleted documents to batch up	When synchronization mode is batch, specify the number of documents to be deleted as a batch.

3 On the **Filters** tab, you can make this setting:

Configuration setting	What to specify
Binary document text filter directory	Specify where to find the Autonomy filters for importing documents that have a binary file format. The default location is the **\Autonomy\OmniSlaves** of exteNd Director's installation directory.

4 Click **Next** to go to the next wizard panel:

- If you selected Typical setup, see "Directory configuration" on page 38.
- If you selected Custom setup, see "Content Management caching configuration" next.

Content Management caching configuration

1 On the Content Management Caching Configuration panel, make the settings you want as follows:

Configuration setting	What to specify
Cache Fields?	Select the types of objects you want to cache. Usually you will want to cache all object types. The reason caching is efficient is that the application makes fewer SQL queries of the database. If there are constraints on memory usage, you may choose not to cache. Content Management caching is not related to DAC cache settings.
Cache Doc Types?	
Cache Folders?	
Cache Categories?	

2 Click **Next** to go to the next wizard panel. See "Directory configuration" next.

Directory configuration

1 On the **Directory Configuration** panel, select a server-specific security realm.

The security realms are as follows:

Realm configuration	Description
LDAP	Base configuration for read and write access to eDirectory™ in exteNd Director. **NOTE:** This realm does not integrate with any supported application server's authentication mechanism.
PersistManager	Read and write access to the user and group repository in the exteNd Director database.
exteNd Server	Read and write access to an exteNd Application Server security provider. The default configuration is SilverUsers.
exteNd ServerLDAP	Read and write access to an exteNd Application Server using the Novell eDirectory LDAP implementation.
exteNd Server (compatible)	Read and write access to a backward-compatible exteNd Application Server. This realm uses groups from the exteNd Director database and users from Silver Security.
WebLogic	Read and write access to WebLogic application server realm APIs, Version 6.x (deprecated). 📖 See WebLogicLDAP below.
WebLogic (readable only)	Read-only access to WebLogic application server realm APIs, Version 6.x., for server cluster environments (deprecated). 📖 See WebLogicLDAP below.
WebLogicLDAP	Read and write access to a WebLogic application server realm using the Novell eDirectory LDAP implementation.
WebSphere	Read and write access to a WebSphere custom registry using the exteNd Director database as a user and group repository. **IMPORTANT:** This realm requires a shared library configuration.
WebSphereLDAP	Read and write access to a WebSphere application server realm using the Novell eDirectory LDAP implementation.

TIP: If you have a user list from an earlier version of exteNd Director on the target server, select **exteNd server (compatible)**.

2 Click **Next** to go to the next wizard panel:

- If you chose an LDAP realm, you need to do more configuration; see "LDAP realm configuration" next.
- Otherwise, see "Framework configuration" on page 41.

LDAP realm configuration

- If you chose an LDAP realm, you need to specify your LDAP configuration options on the Directory Ldap Configuration panel.

  LDAP configuration options are as follows:

LDAP property	Description
Realm	The selected LDAP realm configuration (read-only).
Realm Name	Name used by the realm to access runtime APIs.
Anonymous User	Anonymous principal name.
Administrator	Name used by the realm to access the LDAP server.
Password	Administrator password (see row above).
Administrator Connections	Number of simultaneous administrator connections (or bindings) allowed.
Administrator Conn Wait	Time to wait (milliseconds) for an admin connection to the LDAP server before timing out.
LDAP Host	Host machine and port for the LDAP server.
Use SSL	Check to connect the LDAP server with the Secure Socket Layer (SSL) for data encryption. **NOTE:** If you are using SSL, it is assumed that you have a valid certificate set up on your application server. For details, see your application server documentation.
New User Container	LDAP tree entry for new user registration. This container allows new users to add themselves to the realm without specifying a distinguished name (DN).
User Container DN	Distinguished name (DN) or fully qualified LDAP name of the user container. This defines the search scope for users and groups in the LDAP tree. (See next row.)
User Container Scope	Scope of user entries in the LDAP tree, relative to the User Container (row above). Options are: ◆ object Entries in the user container base level only ◆ onelevel Entries in the user container and one level beneath it in the tree ◆ subtree Entries in the user container and all levels beneath it
User Object Class	User object class. The default value is **inetorgperson**.
Login Attribute	Attribute representing the user login name. **IMPORTANT:** Do not use spaces in this name.
User Membership Attribute	Optional. Attribute representing the user's group membership. **IMPORTANT:** Do not use spaces in this name.

LDAP property	Description
Group Container DN	Distinguished Name (DN) of the group container object.
Group Container Scope	The scope of user entries in the LDAP directory, relative to the group container (see row above). Options are: ◆ object Entries in the group container base level only ◆ onelevel Entries in the group container and one level beneath it in the tree ◆ subtree Entries in the group container and all levels beneath it
Group Object Class	Group object class. The default value is **groupofnames**.
Group Membership Attribute	Optional. Attribute representing the user's group membership. IMPORTANT: Do not use spaces in this name.
Object Attribute	Name of the attribute that specifies the object type in the LDAP tree. IMPORTANT: Do not use spaces in this name.
UUID Auxiliary Class	Auxiliary class that adds the UUID attribute to the user container. This is necessary for accessing the LDAP realm from the exteNd Director APIs.
UUID Attribute	Name of the UUID attribute (see row above). IMPORTANT: Do not use spaces in this name.
Use Dynamic Groups	Check to use dynamic groups.
Dynamic Group Object Class	Dynamic group object class. Default value is **dynamicGroup**.
Dynamic Group Aux Object Class	Auxiliary class that adds the necessary support for accessing dynamic groups in eDirectory. The default value is **dynamicgroupaux**.
Connection Timeout (millis)	Time to wait (milliseconds) for a user connection to the LDAP server before timing out.
Root Container Distinguished Name	Distinguished name of the root (parent) container object—for example, **organization**.
Container Object Types	Lists the current (selected) container object types and attribute names.
Add a new Container Object	Adds support for a new container object. The container must exist within the Root Container hierarchy specified above.

Framework configuration

➢ **To complete the Framework configuration:**

1 On the Framework Configuration panel, specify values for these framework settings:

Framework option	What to specify

Director Framework Datasource	Specify the JNDI name for your application's database. The database contains exteNd Director application data such as content and user information. The value you specify depends on the target application server.
	For exteNd Application Servers
	◆ **To use a connection pool** Replace %CONNECTION_POOL_NAME% with the name of the connection pool you created for the exteNd Director database. For example: if the connection pool is named MyDB, specify **JDBC/MyDB**.
	◆ **To use the deprecated Add Database** The JNDI name follows the pattern:
	`Databases/%DATA_SOURCE_NAME%/Datasource`
	where %DATASOURCE_NAME% is the name of the added database. For example: if your database is named MyDB, replace the suggested value with **Databases/MyDB/DataSource.**
	NOTE: The application database should be different from the deployment database, which typically is the SilverMaster database.
	For application servers other than the exteNd Application Server
	Replace the suggested value with the JNDI name for the database. For example: for a data source named MyDB, specify **MyDB**.
	For any application server
	Typically you will not need to access the JNDI name directly in your applications. But if you need to do so, you can use the Framework API to get the property stored in the FrameworkService config.xml, as shown in this coding technique:
	`// get an EbiConfig object`
	`com.sssw.fw.api.EbiConfig myconfig =`
	`com.sssw.fw.factory.EboFactory.getConfig();`
	`// get value from the key in config.xml`
	`String dsname =`
	`myconfig.getProperty("com.sssw.fw.datasource.jndi-name");`
	`// access the data source`
	`try{`
	`javax.naming.InitialContext ctx = new`
	`javax.naming.InitialContext();`
	`javax.sql.DataSource source =`
	`(javax.sql.DataSource)ctx.lookup(dsname);`
	`}`
	`catch(javax.naming.NamingException ne){}`

Locksmith	To use ACL-based security, specify the user ID for administering security using the DAC. The ID you specify must exist in the server's authentication realm when you deploy the exteNd Director EAR or WAR. **NOTE:** If you do not want to use ACL-based security in your application, set the Locksmith to **anonymous** to prevent ACLs from being set up for the exteNd Director Admin elements. 📖 For more information about the Locksmith user, see the section on subsystem administrators in the *User Management Guide*.

Framework option	What to specify
Enable User Transaction Support?	When checked, your application uses exteNd Director's transaction support. Uncheck this setting if you are deploying to an application server that does not provide JTA capabilities natively or through third-party extensions.

Next you will specify server cluster options.

2 Specify values for these cluster options:

Cluster option	What to specify
Do you want to use clustering?	Select Yes to enable exteNd Director support for clustering. The server you deploy to must already be set up as part of a server cluster.
Host	Specify the URL of the server that will run the exteNd Director Cache Coordinator. 📖 For more information about using the Cache Coordinator, see Chapter 24, "Using the Cache Coordinator". **NOTE:** You must use the exteNd Director installation program to install the Cache Coordinator on the server.
Port	Specify the RMI port number that the Cache Coordinator will listen on. This must be the same value you specify when you install the Cache Coordinator.

The Generated UID field displays an application identifier used for clustering. You cannot change it.

3 Specify a writable directory for exteNd Director use:

Option	Value
Server Accessible Temp Directory	Specify a directory to which the server has write access. The default is the user's TEMP or TMP environment variable. exteNd Director uses the directory for several types of files: ◆ Temporary files ◆ The getmacaddr utility for generating unique identifiers (UIDs) for exteNd Director objects (if getmacaddr doesn't exist in the directory you specify, exteNd Director copies the utility from the FrameworkService JAR to this location) ◆ Parent folder of PortalCache The value you specify is stored in the Framework's config.xml in the key ContentCache.Disk.directory.

4 Click **Next** to go to the next wizard panel. See "AES encryption key" next.

AES encryption key

exteNd Director encrypts portlet preferences in the database using a default key that is **not** unique. You can force the default key to be unique using the FIPS-approved AES encryption. If you require FIPS-compliant security for your application, it is recommended that you have your new projects generate a unique key.

1 On the AES Encryption Key panel, specify the following options:

Option	Value
Generate New AES encryption key?	When checked, the wizard generates a unique encryption key for the stored portlet preferences. Checking this generates a new encryption key file.

2 Click **Next** to go to the next wizard panel:

- If you selected an EAR, see LDAP user options next.
- If you selected a WAR and an LDAP realm, see "LDAP user options" on page 43.

LDAP user options

- If you selected an LDAP realm, you need to specify your user options on the User Ldap Options panel.

The LDAP user options are as follows:

LDAP option	What to specify
Exclude User Attributes	User attributes you want to be inaccessible to the exteNd Director APIs.
Include User Attributes	User attributes you want to be accessible from the exteNd Director APIs.
Include Auxiliary Classes	(Optional) Use to include any custom auxiliary class attributes. These will be added to the user object class hierarchy. Use "|" to separate classes and "," to separate attributes. For example: `auxclass1,attr1,attr2|auxclass2,attr1,attr2`
Exclude Syntax Definitions	Syntax definitions you want excluded from the exteNd Director APIs. LDAP syntaxes determine the data types that can be stored as an attribute. They are defined in RFC 2252 and RFC 2256.

IMPORTANT: Before you deploy a project that uses an LDAP realm, you need to perform these configuration steps:

- Import the UUID auxiliary class provided with the exteNd Director install.
- Set up your LDAP server to use SSL (if applicable).

You can perform these steps after you complete the EAR Wizard.

For details, see the section on "LDAP predeployment tasks (eDirectory only)" on page 191.

Summary panel

1 On the Summary panel, clear the **Build project after wizard is finished** check box if you don't want to build the project archives right away.

TIP: A recently built project is necessary for editing J2EE descriptor files and for deploying exteNd Director Web tier tools. You can let the wizard start the build process or you can select Build commands on the Project menu later.

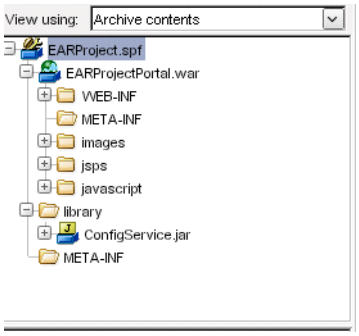
2 Click **Finish** to create the project.

The wizard takes some time to copy the files to the project directory. Then it builds the project if you selected that option.

What the wizard generates

The Project Wizard generates a J2EE-compliant WAR or EAR (depending on your selection). The project contains the subsystem files needed for compile or runtime.

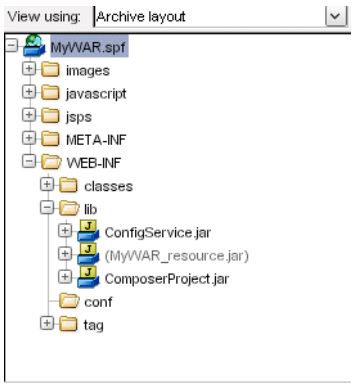
exteNd Director EAR project structure At the top level of an exteNd Director EAR project, you will see:



The EAR project contents include:

File	Description
Portal WAR file	Contains the JARs needed by the exteNd Director subsystems included in the project.
ConfigService.jar	Located in the \library folder. Contains configuration and service properties files for the project's subsystems. 📖 For more information, see "Configuration and service files" on page 45.
JAR files	Located in the \library folder. The set of JARs needed by your application for either compile or runtime services. In a shared library configuration, these service JAR files appear grayed out and in parentheses; they are available for compile-time use but are not included in the deployment archive.

exteNd Director WAR project structure At the top level of an exteNd Director WAR project, you will see:



The WAR project contents include:

File	Description
ConfigService.jar	Located in the WEB-INF\lib folder. Contains configuration and service properties files for the project's subsystems. 📖 For more information, see "Configuration and service files" on page 45.
JAR files	Located in the WEB-INF\lib folder. The set of JARs needed by your application for either compile or runtime services. Files that are grayed out in the display are needed for compiile time only and are not deployed to the server.

Configuration and service files Each exteNd Director subsystem relies on a configuration file and a services file. The files are stored in the ConfigService.jar.

File	Description	Location
config.xml	Sets the configuration properties for a subsystem	Located in the *subsystem-name*-conf subdirectory of the ConfigService.jar For example, the Security subsystem has a config.xml file and a services.xml file in the SecurityService-conf subdirectory
services.xml	Sets properties for each of the services associated with a subsystem	

EAR namespacing The J2EE specification does not require the EAR name to be part of the context for a WAR file. This can be a problem when multiple EAR files with similar content are deployed to the same server. For example: if you deploy the same WAR in several different EAR files to the same server, the application server will not be able to distinguish the different versions of the WAR. To solve this problem, exteNd Director introduces the notion of *EAR namespacing*. The namespace is part of the URL for accessing the exteNd Director Web tiers. exteNd Director prefaces the context for each WAR with the EAR name as the default namespace. The context for each WAR is specified in the application.xml file for the EAR.

NOTE: On a Novell exteNd Application Server: if you deploy to a database other than SilverMaster, the URL would also include the database name.

Classloading within an exteNd Director EAR A WAR file can access classes in a JAR file that is contained within that WAR, as well as classes in a JAR file that is placed outside the WAR in some other location within the EAR. exteNd Director applications take advantage of both of these configurations.

Classpath for WAR When a WAR file uses the services of a JAR file within the WAR, the classes in the WAR have no difficulty locating the classes in the JAR—since these classes are automatically included in the WAR's classloading environment. However, when a WAR file uses the services of a JAR file that is located outside the WAR, the JAR must be added to the *classpath* for the WAR. J2EE applications do this by setting the classpath in the MANIFEST.MF file in the META-INF directory within the WAR.

Classpath for Framework exteNd Director applications also maintain a *simulated classpath* for the Framework, which allows the classes in the Framework to find classes in the other subsystem JAR files that have been included with a particular exteNd Director EAR configuration.

Subsystem architecture

Each exteNd Director subsystem has one or more archive files associated with it. When you select a subsystem in the Project Wizard, the wizard adds the archives you need for the project type you're creating (EAR or WAR). In addition, the wizard automatically enforces any dependencies that exist for the subsystem. If for example you include a subsystem that depends on others, those other subsystems are also included in your project.

Subsystem archive naming conventions The subsystem archive files conform to the following naming convention:

Subsystem archive file	Description
*subsystem-name*Service.jar	Contains the core business objects defining the subsystem, as well as any resource bundles required for internationalization.
*subsystem-name*Tag.jar	Contains a custom tag library that you can use in JSP pages in other Web tiers. A tag library has an associated TLD file that defines the available tags (methods) available to JSP pages.
*subsystem-name*Ejb.jar	Provides EJB support for a subsystem. This JAR contains the server-side implementation required for EJB support. The Project Wizard does not add the EJB JAR files to your project automatically. To use an EJB JAR in your application, you need to use exteNd Director facilities to add the JAR to your project. The EJB JAR files are installed in the **Director\templates\Director** subdirectory of your exteNd Director installation directory. **NOTE:** This file is provided for backwards compatibility. It is not recommended for new development.
*subsystem-name*Client.jar	Contains the EJB client stubs and potentially the Web service stubs needed to access the core business objects of a subsystem. The Project Wizard does not add the EJB client JAR files to your project automatically. To use an EJB client JAR in your application, you need to add the JAR to your project. The client JAR files are installed in the **Director\templates\Director\library** subdirectory of your exteNd Director installation directory. **NOTE:** This file is provided for backwards compatibility. It is not recommended for new development.

Subsystem dependencies The Project Wizard and Setup Wizard enforce subsystem dependencies for nonshared library environments. If you include a subsystem that depends on others, those other subsystems are also included in your project. If you try to remove a subsystem that others depend on, the Setup Wizard forces you to remove the dependent subsystems first before you can remove the subsystem they depend on. In a shared library environment, the wizard cannot enforce any dependencies.

The following table outlines the interdependencies among the exteNd Director subsystems:

To use this subsystem	You need these subsystems	Additional notes
Content Management	◆ Directory ◆ Framework ◆ Search ◆ Security ◆ User	—

To use this subsystem	You need these subsystems	Additional notes
Directory	◆ Framework	The DirectoryServiceRealm.jar file is also required for the Directory subsystem. When you create an exteNd Director EAR project, this JAR file is automatically included in the /library directory of the project. In a WAR project, this JAR file is automatically added to the WEB-INF\lib directory.
Framework	—	—
Pageflow	◆ Directory ◆ Framework ◆ Portal ◆ Portlet ◆ Security ◆ User	—
Portal	◆ Directory ◆ Framework ◆ Portlet ◆ Security ◆ User	—
Portlet	◆ Directory ◆ Framework ◆ Portal	—
Rule	◆ Directory ◆ Framework	PortalCA.jar provides conditions and actions related to resources in the Portal subsystem.
Search	◆ Framework	—
Security	◆ Directory ◆ Framework	—
User	◆ Directory ◆ Framework ◆ Security	—
Workflow	◆ Directory ◆ Framework ◆ Pageflow ◆ Portlet ◆ Security	The Workflow subsystem is loosely coupled with the Rule subsystem. WorkflowAL.jar contains the classes for activities and links provided with exteNd Director.
exteNd Composer Service	◆ Directory ◆ Framework	—

Third-party archive files Several additional third-party archive files are required for the successful execution of exteNd Director applications. In a nonshared library environment, these modules are delivered in the /library directory of an exteNd Director EAR project (the WEB-INF\lib directory of a WAR project). Most manifest files (META-INF\MANIFEST.MF) have these modules listed as dependencies.

3 Reconfiguring exteNd Director Projects

You can change your exteNd Director EAR or WAR by:

- Changing the configuration of various subsystems, including their resource sets
- Adding subsystems from the same or another template
- Updating a project license
- Removing or disabling subsystems
- Changing a project's shared library configuration

Changing the configuration

This section describes ways you can change configuration settings. It includes three topics:

- Changing configuration settings using a wizard
- Changing configuration settings by editing the config.xml file directly
- Changing configuration settings using a predefined view

Changing configuration settings using a wizard

➢ **To change configuration settings using a wizard:**

1 Open your exteNd Director project.

2 Select **Project>Director>Configuration**.

3 In the Project Configuration dialog, click the tab for the subsystem whose settings you want to change:

Subsystem	What you can change	For more information, see
Content Management	Cache options and Autonomy search options	"Content Management caching configuration" on page 38
Directory	Security realm	"Directory configuration" on page 38
Framework	exteNd Director data source, locksmith, and clustering options	"Framework configuration" on page 41
License	exteNd Director project license information	"Updating a project license" on page 52
Pageflow	Resource set binding	"Binding subsystems to a resource set" on page 71
Rule		
Security		

Subsystem	What you can change	For more information, see
User	User service options	“LDAP user options” on page 43
Workflow	Resource set binding	“Binding subsystems to a resource set” on page 71

4 Click **OK** to save your changes.

Changing configuration settings by editing the config.xml file directly

➢ **To change configuration settings by editing the config.xml file directly:**

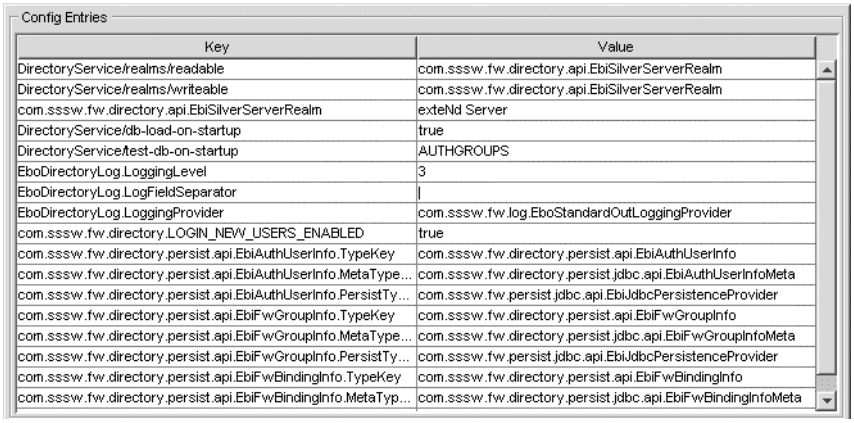
1 Open your exteNd Director project.

2 Locate the config.xml file for the subsystem you want to change.

The config.xml files are located in the project’s ConfigService.jar.

For example, in a WAR the Directory subsystem has a config.xml file in the WEB-INF\lib\ConfigService\ConfigService.spf\DirectoryService-conf subdirectory.

exteNd Director provides a standard key/value editor for config.xml files:



TIP: You can edit the XML source view if you prefer.

In some versions of exteNd Director, the XML source view does not include multiple-line comments. You can open the XML file in a text editor to be sure that you see the complete source.

Changing configuration settings using a predefined view

➢ **To change configuration settings using a predefined view:**

1 Open your exteNd Director project.

2 Go to the **View** tab within the **Resources** tab and select **settings.ear.xml**.

The settings.ear.xml view lets you find configuration settings in an EAR project (in a WAR project, you need to use the **settings.war.xml** view).

The settings view includes a folder containing the configuration files for all subsystems.

Adding subsystems

You can use the Setup Wizard to add subsystems to an existing 5.0 EAR project.

NOTE: This menu option is only enabled when you are working with a Version 5.0 EAR project. It is disabled for all other project types.

You can add subsystems from any template—you don't have to use the same template you used to create the project. The Setup Wizard copies subsystem files to the project directory, adds subprojects to the project file, and makes configuration changes where necessary.

➢ **To add subsystems to your exteNd Director project:**

1 Open your exteNd Director EAR project.

2 Select **Project>Director>Setup**.

3 On the Setup Option panel of the Setup Wizard, select **Add**.

4 On the Template Location panel, specify the directory for the template that contains the subsystems you want to add, then click **Next**.

📖 For information about selecting templates, see "Creating an exteNd Director project" on page 33.

5 Click **Yes** to confirm your template choice.

6 On the Subsystem Selection panel, select the subsystems you want to add.

The wizard displays panels for configuration options.

7 Make configuration settings as appropriate.

"Creating an exteNd Director project" on page 33 has information about these panels (for example: see "Content Management caching configuration" on page 38, "Content Management caching configuration" on page 38, and "Summary panel" on page 43.)

8 On the last panel, check the summary information and optionally clear the **Build project after wizard is finished** check box.

9 Click **Finish**.

Just as with the wizard for creating the exteNd Director project, this wizard takes some time to copy subsystem files from the template to the project directory. Then it builds the project, if you selected that option. When it finishes, you're ready to continue work on your application.

Removing or disabling subsystems

NOTE: For 5.0 EAR projects only.

The Setup Wizard lets you remove or disable any installed subsystem.

Removing means the subsystem is removed from the project and references are removed from the project definition. Its files are deleted from disk.

Disabling means the subsystem files remain part of the project, but the subsystem is omitted from the archive when the project is built. The subsystem files are still in the project and you can reenable it when you want to.

In both cases, references to the subsystem are removed from configuration files and classpaths in manifest files.

➢ **To remove subsystems from your exteNd Director project:**

1 Open your exteNd Director EAR project.

2 Select **Project>Director>Setup**.

3 On the Setup Option panel of the Setup Wizard, select **Modify**.

4 On the Subsystem Setup panel, clear the **Selected** check box for a subsystem you want to remove. Click **Yes** when the wizard asks if you are sure, warning that its files will be deleted.

TIP: If you select a subsystem that another subsystem depends on, the wizard won't allow it to be deleted. A message tells you what subsystems depend on it.

The wizard deletes the subsystem from the project and the subsystem files from disk.

5 Delete additional subsystems if you want, and click **Finish** when you are done.

➢ **To enable or disable subsystems:**

1 Open your exteNd Director project.

2 Select **Project>Director>Setup**.

3 On the Setup Option panel of the Setup Wizard, select **Modify**.

The wizard displays the Subsystem Setup panel.

4 Disable a subsystem by clearing its **Enabled** check box. Then click **Yes** when the wizard asks if you are sure you want to disable the subsystem.

The wizard deletes the subsystem from the project but leaves the files on disk so that you can reenable the project later.

5 Enable a subsystem by selecting its **Enabled** check box.

The wizard restores the subsystem to the project.

6 Disable or enable additional subsystems if you want, and click **Finish** when you are done.

➢ **To find out what subsystems are enabled:**

1 Open your exteNd Director project.

2 Select **Project>Director>Information**.

The Subsystem Information dialog displays the version and enabled status of the subsystems in your project.

3 Click **OK** when you are done.

Updating a project license

exteNd Director evaluation and beta software includes a project license that expires after a certain number of days. This means that you will not be able to create new exteNd Director projects or access deployed exteNd Director applications (built using that license) after it expires. You'll know when your project license has expired when you get an error message such as:

Occurence	Error message
Design time	License string *nnnn* has expired
Run time	Maximum license count 0 exceeded

To update an evaluation or beta license for new and existing projects, you'll need to:

1 Obtain a new license.

2 Update the template license.

3 Update the license in each existing project that you want to continue to use.

Obtaining a new project license You can obtain a new project license by:

- Purchasing an unrestricted license and updating the template license as described in the procedure "To update the template license:" on page 53. Contact your sales representative for information about obtaining this type of license.
- Purchasing and installing a commercial version of exteNd Director.

  If you have existing exteNd Director projects that you want to continue to use (and they were created using the beta or evaluation software) you'll need to update those projects as described in the procedure "To update the template license:" on page 53).

Updating the license The exteNd Director project license is stored in the default project template and in each individual project.

To update this project license	See this procedure
Existing projects that I want to continue to use	"To update a project with expired licenses:"
Template license for all new projects	"To update the template license:"

If you have existing exteNd Director projects, you'll need to update each project individually. Follow the steps

➢ **To update the template license:**

1 Open an existing or create a new exteNd Director project.
2 Choose **Project>Director>Configuration**.
3 Choose **License**.
4 Type or paste the new license in the License String field.
5 Click **Update License template**.

  The currently open project is updated with the new license and any subsequent exteNd Director projects that you create will include this new license.
6 Click **OK**.

➢ **To update a project with expired licenses:**

1 Open an existing or create a new exteNd Director project.
2 Choose **Project>Director>Configuration**.
3 Choose **License**.
4 Type or paste the new license in the License String field.
5 Click **OK**.
6 Rebuild the project.
7 If the project was deployed, redeploy it.

Changing a project's shared library configuration

exteNd Director relies on JARs to provide subsystem resources. For example, each subsystem has one or more archives associated with it. In addition to the subsystem JARs, exteNd Director applications require other third-party JARs for successful execution. Examples of these other JARs include: Xalan.jar and xercesImpl.jar.

You can decide how exteNd Director should package the JARs used by your projects. The JAR configuration options are described in the following table:

If you want to	Choose this configuration
Include all of the needed JARs within your project	This is called a *nonshared library* configuration. By default, the Project Wizard creates all exteNd Director projects in this configuration. 📖 For more information, see "About nonshared library configurations" on page 54.
Include only the exteNd Director subsystem JARs within the project . Copy the third-party JARs to a well-known location on the deployment application server	This is called a *3rd party JARs* (partial shared library) configuration 📖 For more information, see "About 3rd party JAR configurations" on page 55.
Don't include any framework or subsystem JARs within the project. Copy all of the framework, subsystem and 3rd party JARs to a well-known location on your deployment application server	This is called a *shared library* configuration. You **must** use this configuration: ◆ When you want to run standalone portlet applications. ◆ When you deploy to an IBM WebSphere application server using the WebSphere custom realm. 📖 For more information, see "About shared library configurations" on page 55

About nonshared library configurations

By default, exteNd Director applications (except for portlet applications) are created using the nonshared library configuration.

In the *nonshared library configuration*:

- Each exteNd Director application is a self-contained unit.

  All of exteNd Director's services are embedded within an exteNd Director EAR or WAR project. The application server does not share any files among the exteNd Director applications that are deployed.
- The classes are loaded by individual J2EE WAR or EAR class loaders.
- You can deploy multiple portals to the application server, but they must use different exteNd Director databases.

You might **not** choose the nonshared library environment because:

- The project size is large resulting in a longer deployment time.
- Portlet application WARs must be included in an exteNd Director EAR for deployment. In the nonshared library configuration, portlet application WARs cannot be deployed independently.
- Changes to the application require a redeployment of the archive.

📖 For more information on changing to a different configuration, see "Procedures for changing the project configuration" on page 56.

About shared library configurations

In a *shared library* configuration, the JAR files and classes that provide exteNd Director's services are installed on your application server in a well-known location. This means that the JARs can be shared across all exteNd Director Web applications deployed on that application server. The classes in the shared libraries are loaded by a single class loader, and the Web application class loaders extend from that class loader.

The shared library's benefits and restrictions are outlined below:

Benefits	Restrictions
Smaller project size	The application server is limited to a single deployed portal
Faster deployment	Configuration changes and redeployments require an application server restart
Applications can be independently redeployed	You cannot use the rapid deployment feature.
Decouples exteNd Director server configuration information from client application	—
You can deploy portlet application WARs independently	—

📖 For more information on changing to a different configuration, see "Procedures for changing the project configuration" on page 56.

About 3rd party JAR configurations

In the *3rd party JAR* configuration:

- The JAR files and classes that provide exteNd Director's services are included in in the project.
- The 3rd party JARs are installed on your application server and thus can be shared across all exteNd Director Web applications deployed on that application server.

The benefits to the 3rd party JAR configuration include:

Benefits	Restrictions
Smaller project size	—
Faster deployment	Configuration changes and redeployments require an application server restart
Applications can be independently redeployed	You cannot deploy portlet application WARs independently
You can deploy multiple portals to the application server, but they must use different exteNd Director databases.	—

📖 For more information on changing to a different configuration, see "Procedures for changing the project configuration" on page 56.

Procedures for changing the project configuration

By default, exteNd Director projects use a nonshared library configuration.

➢ **To determine the current project configuration:**

1 Open the project.

2 Choose **Project>Director>Shared Lib**.

The Shared Lib dialog displays.

If the Shared Lib check box is not selected, the project is a nonshared library project.

➢ **To determine the application server's configuration:**

- To determine your application server's configuration, you'll need to check the appropriate server directory and see which JARs are there (or not).

Server name	Check this location
Novell exteNd Application Server	**File:** AgJars.conf **Directory:** extend5\sharedlib
Apache Tomcat	**Directory:** shared\lib
BEA WebLogic	The server's classpath
IBM WebSphere	**Directory:** server\lib

If the location contains:

- JARs (like jaxrpc-api.jar, jdom.jar, js.jar, and log4j.jar), but none of exteNd Director's subsystem JARs, then it is a 3rd party shared library configuration.
- exteNd Director's subsystem JARs (like FrameworkService.jar, RuleService.jar, or DirectoryService.jar), then it is a full shared library configuration.
- Neither 3rd party nor exteNd Director's subsystem JARs, then it is a nonshared library configuration.

➢ **To change a nonshared library server to a full or 3rd party shared library configuration:**

1 Undeploy any exteNd Director projects that require a nonshared library environment.

2 Stop your server.

3 Open a project.

NOTE: This procedure also changes the structure of the project.

4 Choose **Project>Director>Shared Lib**.

The Shared Lib dialog displays.

5 Click **Shared Lib** check box.

- To change from nonshared library to 3rd-party JARs only, click both **Shared Lib** and **Only 3rd party JARs**).

  You'll see the list of JARs that will be removed from the project and added to the application server. The list varies depending on your selection.

6 Click **Copy JARs**.

You are prompted for a directory location to copy the JARs to.

6a Browse to the server's directory listed in the following table:

Server name	Check this location
Novell exteNd Application Server	**File:** AgJars.conf **Directory:** extend5\sharedlib
Apache tomcat	**Directory:** shared\lib
BEA WebLogic	The server's classpath
IBM WebSphere	**Directory:** server\lib

6b Click **OK**.

7 Update the application server's classpath so that the server can locate the JARs you just copied.

- For Novell exteNd Application Servers, you can click **Update AgJars.conf** to automatically update the application server's classpath.
- For other servers, see their documentation for information about updating the classpath.

8 Restart your server.

➢ **To change from a shared library server configuration to a nonshared library configuration:**

1 Undeploy any exteNd Director projects that require a shared library environment.

2 Stop your server.

3 Open a project.

NOTE: This procedure also changes the structure of the project.

4 Choose **Project>Director>Shared Lib**.

The Shared Lib dialog displays.

5 Unselect the **Shared Lib** checkbox and (if checked) the **Only 3rd party JARs**.

6 Click **OK**.

7 Remove the exteNd Director subsystem JARs and the 3rd party JARs from the server's directory.

Server name	Check this location
Novell exteNd Application Server	**File:** AgJars.conf **Directory:** extend5\sharedlib
Apache tomcat	**Directory:** shared\lib
BEA WebLogic	The server's classpath
IBM WebSphere	**Directory:** server\lib

8 Remove the exteNd Director subsystem JARs and the 3rd party JARs from the server's classpath.

9 Restart your server.

4 Updating exteNd Director Projects

The exteNd Director Setup Wizard provides a way to update EAR project files. It compares the files in a template (normally the template from which the project was created) to the files in your project directory, then copies all files that are not present or have changed. This chapter contains one section:

- Procedure for updating your exteNd Director project

NOTE: This update utility does not analyze how your project is configured. It copies files from subsystems that your project does not include.

Procedure for updating your exteNd Director project

NOTE: This menu option is only enabled when you are working with a Version 5.0 EAR project. It is disabled for all other project types.

➢ **To update your exteNd Director project:**

1. Open your exteNd Director project.
2. Select **Project>Director>Setup**.
3. On the Setup Option panel of the Setup Wizard, select **Update**.
4. In the **From location** text box, enter the location of the source project, typically the standard EAR template in your exteNd Director installation. For example:

   ```
   C:\Program Files\Novell\exteNd5\Director
   ```

5. In the **To location** text box, enter the location of the root of your target EAR project. For example:

   ```
   C:\Director_Projects\My_Project
   ```

6. Select the **Mode**:

Mode	What it means
All	Update all files (no filtering)
Classes	Update class files only (intended for updating subsystem classes in the library WAR)
Selective	See Step 7 (next)

7. If you chose **Selective**, specify lists of extensions in the **Include Extensions** and/or **Exclude Extensions** fields. Separate extensions by semicolons.
8. Check **Check date & length** to replace files that have a different length in addition to those that have a different date.
9. (Required for version updates) Check **Copy new files** to include new files in addition to updated files.

   Any file that exists in the **From location** but not in the **To location** is considered to be a new file.
10. Check **Trial** to see a list of the files to be replaced or copied without actually performing the update.
11. Click **Go**.

5 Working with exteNd Composer Projects

This chapter explains how to add new and existing exteNd Composer projects as subprojects of your current project. It contains the following sections:

About exteNd Composer projects

Before you run the Composer Pageflow Wizard, you need to add an exteNd Composer subproject to your exteNd Director project. You can do this in either of two ways:

- By creating a new exteNd Composer project
- By adding an existing exteNd Composer project

Once you've added the exteNd Composer subproject to your exteNd Director project, you need to deploy your exteNd Director project at least once. Then you can begin to take advantage of the vulturing capabilities provided by the resource set (since the exteNd Composer subproject is added to the resource set). The resource set ensures that any change you make to an exteNd Composer project artifact is automatically picked up by the server and can be tested right away. This is also true of the pageflow, since pageflows are also stored in the resource set as well.

NOTE: The resource set being used by exteNd Composer must be the same one used by the pageflow. If they are not the same, the pageflow will not be able to find all of the resources associated with the service.

For complete details on working with exteNd Composer services, see the exteNd Composer help. For details on creating exteNd Composer pageflows, see the chapter on using the Composer Pageflow Wizard in the *Pageflow and Form Guide*.

Working in the Express Portal project The Express Portal project contains a exteNd Composer subproject. Therefore, if you plan to work in the Express Portal project, you do not need to add an exteNd Composer subproject.

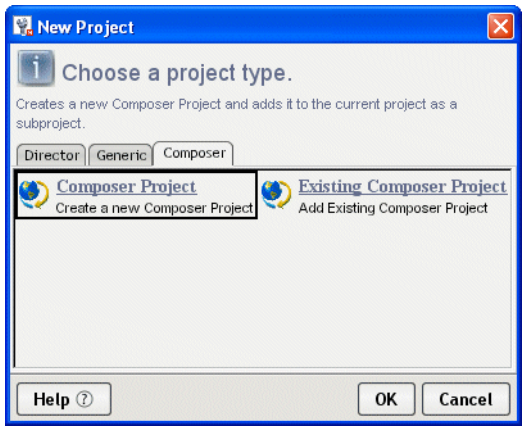
Creating new exteNd Composer projects

➢ **To create a new exteNd Composer project within an exteNd Director project:**

1. In the exteNd Director development environment, open the exteNd Director project that will contain the exteNd Composer subproject.
2. Select **File>New>Project**.

   The New Project dialog appears.

3 Select the **Composer** tab and choose **Composer Project**:



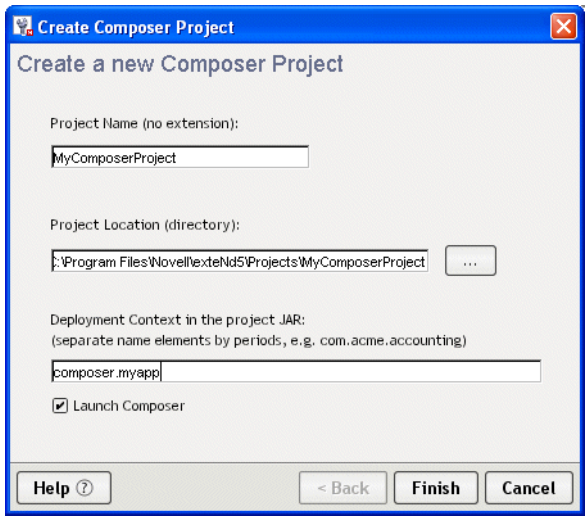
4 (Required) Type a project name. exteNd Director adds the project name extension (.spf).

5 Specify the directory where you want your project to reside. Select **...** to browse the directories on your computer.

6 Enter a deployment context string in the bottom-most text field of the dialog. The string should contain labels (no spaces) separated by periods, as in `composer.myapp`.

NOTE: The context string should not contain Java-language keywords, such as `try`, `catch`, `finally`, `int`, `for`, and so forth.

7 To launch exteNd Composer after the project is created, leave the **Launch Composer** checkbox selected:



8 Click **Finish**. The window appears with the name of the project you just created in the title bar.

Adding existing exteNd Composer projects

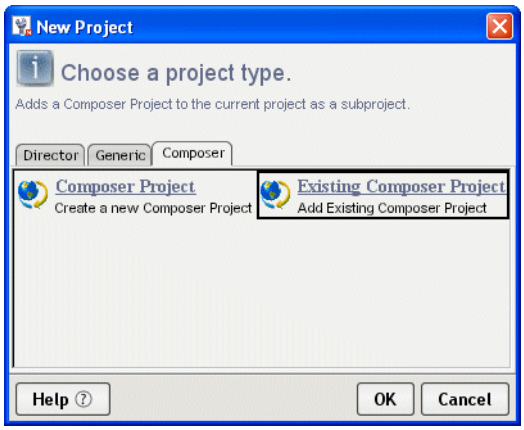
➢ **To add an existing exteNd Composer project to an exteNd Director project:**

1 In the exteNd Director development environment, open the exteNd Director project that will contain the exteNd Composer subproject.

2 Select **File>New Project**.

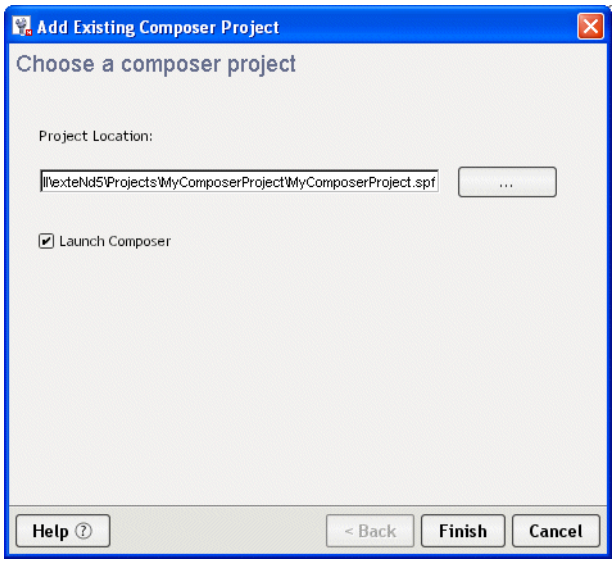
The New Project dialog appears.

3 Select the **Composer** tab and choose **Existing Composer Project**:



4 Specify the directory where you want your project to reside. Select **...** to browse the directories on your computer.

5 To launch exteNd Composer after the project is created, leave the **Launch Composer** checkbox selected.



6 Click **Finish**. The window appears with the name of the project you just created in the title bar.

III Managing Application Resources

Explains how to use the resource set to manage application resources.

6 Using the Resource Set in an exteNd Director Application

This chapter describes the purpose of the exteNd Director resource set, what it contains, and how its configuration settings affect your exteNd Director application. It contains the following sections:

Role of a resource set in your application

An exteNd Director *resource set* organizes descriptors and other files used by exteNd Director subsystems and provides for dynamic loading during development, avoiding frequent redeployments and speeding up your testing. Each Portal (or Portlet) application can include a resource set.

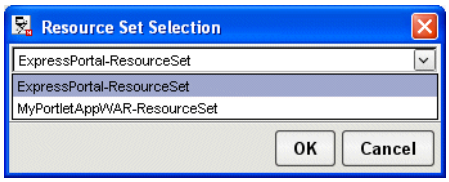
A resource set holds application-defined resources and classes. Some of these resources are templates or definitions for using a subsystem's features, such as a workflow or a portlet descriptor. Others specify how subsystems work together, such as bindings between rules and users. Resources are usually XML files; some are accompanied by Java classes.

Support for multiple resource sets The exteNd Director subsystems provide support for multiple resource sets. Therefore, an exteNd Director EAR that contains multiple portlet application projects, each with a separate resource set. Each portlet application resource set can have its own resources (such as pageflows and workflows). These resources do not need to be copied into the resource set for the Portal WAR.

Each portlet application WARs (whether in shared lib mode, or WARs contained within an EAR) is a self-contained unit. A full portlet application can therefore be deployed without requiring a change to the Director deployment.

Artifacts in a WAR can only access other artifacts in the same WAR. This restriction reinforces the notion that each application WAR is a self-contained entity.

When you create a new resource in an EAR project that contains multiple resource sets, you need to specify which resource set to use as the target.



Finding resources A resource set organizes your application's resources in a known directory structure, described in "What to put in a resource set" on page 68.

In your application, access to resources is handled by the *resource servlet*. These are its primary functions:

- Document discovery and retrieval
- Java class discovery
- Java class loading

Dynamic loading A resource set can be configured to *dynamically load* resources from disk as well as from the deployed WAR. Configuration settings specify where to look for updated versions of resources. The resource servlet *vulture* keeps watch on the directory locations and determines when new classes and resources are available to be loaded. To set up dynamic loading, see "Dynamic loading of resources and classes" on page 77.

Resources tab When you are developing your application, you can use the Resources tab in the Navigation Pane to find the resources you want to work on. For more information, see Chapter 9, "Searching a Resource Set".

Including a resource set in your exteNd Director application In an exteNd Director EAR project, a resource set is part of an application WAR within the EAR. In an exteNd Director WAR project, the resource set is added as a JAR file in the WEB-INF\lib directory. A resource set is required for a WAR project—but is not required for an EAR project.

What to put in a resource set

When you create a Portal (or Portlet) application that includes a resource set, the WAR contains the resource set servlet and a JAR file called ***appname*_resource.jar** that contains directories for the resources required by your exteNd Director application. The resource JAR is located in the WEB-INF\lib directory of the WAR.

The resource JAR contains Java classes for portlets, conditions, and actions you create, as well as XML descriptor files that provide application metadata. The resource JAR can also contain custom resources that you define. When you use exteNd Director to create new resources of various types, the exteNd Director wizards save the resources in the appropriate directories of the resource JAR.

In addition to ***appname*_resource.jar**, you can add other JAR files to WEB-INF\lib. Any resources in those additional JARs must be stored in the subdirectory that corresponds with the resource type. Each JAR needs to be listed in the resourcePath and/or libPath in the resource set's configuration file.

Subdirectories for resources and Java classes

The following table lists the directories in a resource JAR and the types of resources they can contain. When you look at the project in Source Layout, you will find these subdirectories under the **data** directory:

Resource subdirectory	Purpose of resources	Tools for creating
form	XTHML Web forms that are XForms-compliant	Form Designer, Database Pageflow Wizard, Composer Pageflow Wizard, and Web Service Pageflow Wizard
framework-database	SQL files for loading data into the framework database	Any text editor
html	HTML pages	HTML File Wizard
images	Graphics files	Any commercially available tool for creating graphics files
my-views	Search queries for changing the set of files you are working with as you develop your application	Resource Set tab
pageflow-process	Pageflow process descriptor	Pageflow Modeler, Database Pageflow Wizard, Composer Pageflow Wizard, and Web Service Pageflow Wizard
portal-category	Label for categorizing portlets and pages. Used in the Portal Personalizer	Category Wizard
portal-component	Component descriptors, which provide configuration information for component classes	—
portal-data-definition	Wireless configuration information	Transcoding Definition (Wireless) Wizard
portal-device-profile	Definitions of user environments. Used in portal-data-definition and portal-style resources. Several are provided.	Device Definition (Wireless) Wizard
portal-layout	Descriptors and definitions of the way a portal page arranges portlets on a page	Layout and Layout Definition Wizard
portal-option	Descriptors for action items that you can include in the title bars of portlets	Option Wizard
portal-page	PID definitions, which are pages that contain tags that display portlets and components. PID pages are processed by the portal servlet.	Page Descriptor Wizard
portal-portlet	Portlet descriptor	Portlet Wizard, Pageflow Modeler, Database Pageflow Wizard, Composer Pageflow Wizard, and Web Service Pageflow Wizard
portal-style	Portal styles (XSL) and portal style descriptors (XML)	Style Descriptor Wizard

Resource subdirectory	Purpose of resources	Tools for creating
portal-theme	Subdirectories that contain files that define the visual characteristics to be applied across a portal application	Theme Wizard
rule	Rule definitions	Rule Designer
rule-action-macro	Action macro definitions	Action Macro Wizard
rule-condition-macro	Condition macro definitions	Condition Macro
rule-group-binding	Associations between rules and groups	Group Bindings Wizard
rule-pipeline	Pipeline definitions	Pipeline Wizard
rule-pipeline-binding	Associations between rules and pipelines	Pipeline Bindings
rule-user-binding	Associations between rules and users	User Bindings Wizard
security-role	Associations between roles and users	XML Editor (see the chapter on role-based authorization in the *User Management Guide*)
workflow-activity-policy	URL representing the client to open for a workitem	XML Editor
workflow-process	Definition of a workflow process	Workflow Process Wizard
wsdl	WSDL (Web Services Description Language) files that describe Web Services	WSDL Editor
xsl	XSL files	XSL Editor
custom-directory-name	Additional directories that contain Java classes or your own custom resources	—

Using views to find what you're looking for You can use *views* to display personalized lists of items within an exteNd Director project. Views can be used to look at resources in a resource set. exteNd Director ships with several predefined views. In addition, exteNd Director allows you to define custom views to display project items that are of particular interest to you. For details on using views to find items in an exteNd Director project, see Chapter 10, "Working with Views".

Projects for a resource set

A Portal (or Portlet) application with a resource set consists of at least two projects, which you see in the Source Layout:

- A WAR project
- A JAR project in the WEB-INF/lib directory of the WAR

The web.xml descriptor for the WAR projects configures the resource servlet. The JAR project has a data directory and a source directory that are used to build the resource JAR. Additional resource JARs can be included as built JARs or as projects whose JARs are built with the current project.

Binding subsystems to a resource set

How binding works Subsystems that use resource sets are bound to them by entries in the following XML files:

- **resourceset.xml** in the WEB-INF/conf directory of a WAR
- **config.xml** for the subsystem service JAR

The resourceset.xml file specifies a name that other modules use to refer to the resource set. You can change the name by editing the value of this setting:

```
<settings>
   <name>appname-ResourceSet</name>
   ...
</settings>
```

For information about resourceset.xml, see "Configuring the resource set" on page 71.

For a subsystem that uses resources, its config.xml file binds the subsystem to a particular resource set by specifying the resource set name in a property key/value pair. The binding for the rule subsystem looks like this:

```
<property>
  <key>RuleService/resourceset</key>
  <value>appname-ResourceSet</value>
</property>
```

Editing the binding There are two ways to edit the binding:

- Use the Configuration tool (Project>Director Project>Configuration)

  For information, see Chapter 3, "Reconfiguring exteNd Director Projects".

  OR
- Edit a subsystem's config.xml

TIP: Use the View tab on the Resources panel in the Navigation Pane to find configuration files. Try these views:

- settings.war.xml for a WAR project (or settings.ear.xml for an EAR project)
- config.services.by.subsystem.war.xml for a WAR project (or config.services.by.subsystem.ear.xml for an EAR project)

Binding and the Project Wizard When you use the Project Wizard to create a new Web application that includes a resource set, the wizard sets the subsystem bindings to point to the new resource set. You use the methods just described to reset them, if you want.

Configuring the resource set

A Web application with a resource set has two configuration files:

- web.xml
- resourceset.xml

About web.xml The web.xml descriptor contains the standard settings for a WAR. It identifies the servlets that the resource set uses. It doesn't hold any resource set configuration settings.

About resourceset.xml The resourceset.xml configuration file has settings that specify how to find resources and what JARs are enabled, as well as variables that you can use when setting values.

The rest of this section describes the settings you can make in resourceset.xml, showing the XML. In the Resource Set Editor, you can use the graphical view so you don't have to edit the XML directly.

For information on using the Resource Set Editor, see Chapter 7, "Editing the Configuration of a Resource Set".

Variables

The variables section of resourceset.xml defines local variables that you can use (instead of static values) when defining configuration settings. Variables can be used to identify whether a subsystem is installed and active. Variables can also be used for directory paths. In a newly created resource set, several variables are defined for you. You can add additional variable definitions.

The variables section has this XML format:

```
<variables>
   <variable key="EARLOCATION" value="C:\DirectorProjects\Test2\Ear" />
   <variable key="WARLOCATION" value="C:\DirectorProjects\Test2\Ear\MyApp" />
   <variable key="ACCESS_DISK" value="true" />
   <variable key="LIBRARY" value="../library" />
   <variable key="WEBINF" value="WEB-INF/lib" />
   <variable key="NEVER" value="0" />
   <variable key="FREQUENT" value="7500" />
   <variable key="INFREQUENT" value="15000" />
</variables>
```

NOTE: The EARLOCATION variable is not included in WAR projects.

These variables are defined for you:

Variable	Typical value	Purpose
EARLOCATION (EAR projects only)	*drive:\project-directory*\EAR	The path for the EAR in the project directory; useful for specifying disk locations in the resourcePath and libPath
WARLOCATION	*drive:\project-directory*\EAR*appname* (EAR projects) *drive:\project-directory\appname* (WAR projects)	The path for the WAR that contains the resource set
ACCESS_DISK	true	Whether resources and classes are being dynamically loaded from disk; should be coordinated with the **vultures** and **dynamicClassLoading** settings for more readable entries in resourcePath and libPath
LIBRARY	../library	The relative path to the library directory, which contains all the subsystem JARs
WEBINF	WEB-INF/lib	The relative path to the directory in the resource set WAR that contains resource JARs
NEVER	0	Variable for setting an entry's vultureInterval attribute
FREQUENT	7500	Variable for setting an entry's vultureInterval attribute
INFREQUENT	15000	Variable for setting an entry's vultureInterval attribute

General settings

General settings for the resource set include its name and flags that enable validation, logging, and dynamic loading.

The settings section of resourceset.xml has this format:

```
<settings>
   <name>appname-ResourceSet</name>
   <dynamicClassLoading>true</dynamicClassLoading>
   <validate>false</validate>
   <verbose>false</verbose>
   <vultures>true</vultures>
</settings>
```

These general settings are as follows:

Element	Typical value	Purpose
name	*appname*-ResourceSet	A name for the resource set; used in other configuration files that need to refer to this resource set.
dynamicClassLoading	true or false	Whether Java classes are dynamically loaded when they are changed. The vultures setting must also be enabled. 📖 For information, see "Dynamic loading of resources and classes" on page 77.
validate	true or false	Whether validation classes in the resource set should run when the resource set is loaded. Typically set to true during development and false in a deployed production application. 📖 For information, see "Validating a resource set" on page 81.
verbose	true or false	Whether log messages are reported to the server console.
vultures	true or false	Whether exteNd Director sets up processes to report changed files in disk locations in the resource set's paths. 📖 For information, see "Dynamic loading of resources and classes" on page 77.

Types and locations of resources: resourcePath and libPath

The **path-entries** section of resourceset.xml specifies two paths: resourcePath and libPath.

resourcePath The **resourcePath** tells the exteNd Director application where to find resources. For resourcePath, you specify:

- What types of resources to load. A set of **ext** elements identify the file extensions associated with resource types.
- Where to find resources. A set of **entry** elements identify JARs and disk locations that contain resources. Resources may be dynamically loaded from disk locations and reloaded when they change.

libPath The **libPath** tells the application where to find Java classes. The resource set class loader looks for classes in particular locations and can dynamically load and replace classes that have been loaded previously. For libPath, you specify:

- What classes are loaded via the normal class loader. A set of **filter** elements identify the packages that contain these classes, typically the packages of the exteNd Director API.
- What file extensions identify Java code—typically just **.**class.
- Where to find Java code in JARs and disk locations. A set of **entry** elements identify JARs and disk locations that contain Java classes. Classes may be dynamically loaded from disk locations and reloaded when they change.

NOTE: Classes you want to load from disk locations must not be included in the EAR or WAR. When you use dynamic classloading, you need to make the projects that include the classes inactive.

Example The path-entries section has this XML format:

```
<path-entries>
   <resourcePath>
      <exts>
         <ext active="true">.xml</ext>
         ...
      </exts>
      <entries>
         <entry active="true">$WEBINF$/RuleCA.jar</entry>
         ...
         <entry active="!$vultures$">$WEBINF$/appname-resource.jar</entry>
         <entry active="$vultures$" vultureInterval="$FREQUENT$"
                recursive="true">$WARKLOCATION$/data</entry>
         ...
      </entries>
   </resourcePath>

   <libPath>
      <filters>
         <filter active="true">com.sssw.fw</filter>
         ...
      </filters>
      <exts>
         <ext active="true">.class</ext>
      </exts>
      <entries>
         <entry active="true">$WEBINF$/CQA.jar</entry>
         ...
         <entry active="!$vultures$">$WEBINF$/appname-resource.jar</entry>
         <entry active="$vultures$" vultureInterval="$FREQUENT$"
                recursive="true">$WARLOCATION$/build/resource-classes</entry>
         <entry active="!$productionMode$" vultureInterval="$FREQUENT$"
                recursive="true">$DISKLOCATION$/ResourceSet/WEB-INF/lib</entry>
      </entries>
   </libPath>
</path-entries>
```

The table describes the elements in the path-entries section:

Element	Purpose
resourcePath	Container for **exts** and **entries** elements that specify what resources to load and where to find them.
libPath	Container for **filters**, **exts**, and **entries** elements that specify what Java classes to load and where to find them.

Element	Purpose
filter	Specifies a class or package containing classes that the normal class loader loads, including classes of the exteNd Director API. Individual **filter** elements are contained in a **filters** container element. If the active attribute is false, the item is ignored. Example: `<filter active="true">com.sssw.fw</filter>`
ext	Specifies a file extension that identifies what files to load from locations on the resourcePath or libPath. Individual **ext** elements are contained in an **exts** container element. If the active attribute is false, the item is ignored. Example: `<ext active="true">.xml</ext>`

entry	Specifies a JAR or disk location where resources or Java classes are found. It is typical to use variables to identify locations within the exteNd Director project. Individual **entry** elements are contained in an **entries** container element.
	The element's data value is a path:
	◆ For **JARs**, the path is the location of a Java archive in the resource set WAR.
	◆ For **disk locations**, the path should be the location of the files in the source layout view of the project.
	Order of entries The entry elements are scanned from last to first. If duplicate resources or classes exist, the location listed **last** is the one that gets used. However, if a class is in the EAR or WAR but also on disk, the class in the EAR or WAR is always used.
	Attributes If the active attribute is false, the item is ignored. Additional attributes that apply when the entry is a disk location and dynamic loading is enabled are:
	◆ **vultureInterval** Milliseconds between scans for updated classes or resources
	◆ **recursive** Whether subdirectories are included
	📖 For more about dynamic loading, see "Dynamic loading of resources and classes" on page 77.
	Examples:
	`<entry active="true">$WEBINF$/CQA.jar</entry>`
	`<entry active="true">$WEBINF$/resource.jar</entry>`
	`<entry active="true" vultureInterval="$FREQUENT$" recursive="true">$WARLOCATION$/data</entry>`
	`<entry active="true" vultureInterval="$FREQUENT$" recursive="true">$WARLOCATION$build/resource-classes</entry>`

Using variables in entry elements

Several useful variables are defined for you in the variables section, including the disk location of the EAR and WAR. You can also use general settings as variables.

📖 For a list of general settings, see "General settings" on page 73. For a list of variables, see "Variables" on page 72. You can define additional variables as needed to make your resource.xml dynamically configurable.

Editing tips When editing XML and paths, include a $ before and after the name of a variable or general setting. In the Resource Set Editor, right-click check boxes to select from a list of variables.

Examples This entry refers to the resource.jar in the resource set. $WEBINF$ specifies the WEB-INF/lib directory of the resource set WAR:

```
<entry active="true">$WEBINF$/resource.jar</entry>
```

This entry refers to a disk location within the exteNd Director project directory where Java classes are compiled. A vulture interval has been set—so as items are recompiled, they will be dynamically loaded:

```
<entry active="true" vultureInterval="$FREQUENT$"
 recursive="true">$WARLOCATION$/build/resource-classes</entry>
```

This entry refers to the disk location where resources are stored in the WAR. The subdirectories will all be searched recursively—and as resources change, they will be dynamically loaded:

```
<entry active="$vultures$" vultureInterval="$FREQUENT$"
 recursive="true">$WARLOCATION$/data</entry>
```

Directory keys for indexing

exteNd Director uses indexes of the resource set content to figure out what to display in the Resource Set tab of the Navigation Pane. Both the Relationship Viewer and the Resource Set Search provide useful ways of looking at your project's content, and the **directories** section of resourceset.xml defines ways for looking at the content.

Files in the resource JARs and disk locations are always indexed by their file names. For each of the standard resource set directories, resourceset.xml can define additional ways of categorizing the resources or classes. You can define additional search indexes, and you can add indexes for custom directories in the resource set.

Indexing and searching are features that enhance your work environment; they are not used in your deployed exteNd Director application.

The directories section has this XML format:

```
<directories>
   <directory name="rule-condition-macro" active="true">
      <search key="name" valuebased="true" xpath="/conditionmacro[@name]" active="true" />
   </directory>
   ...
</directories>
```

The table describes the elements contained in the directories section:

Directory setting	Typical value	Purpose and attributes
directory	`<directory name="rule-condition-macro" active="true">`	The directory for which you want to add indexes. There can be a directory entry for any directory that occurs in any of the JARs or disk locations in the path-entries section. Attributes are: ◆ **name** The name of the directory to be indexed. ◆ **active** Whether this directory should be indexed. For efficient performance, set to false when the appropriate subsystem is not loaded.

Directory setting	Typical value	Purpose and attributes
search	`<search key="name" valuebased="true" xpath="/conditionmacro[@name]" active="true" />`	A definition for a secondary index for the directory. There can be one or more search indexes for a directory. The file name of the resource or class is always the primary index. Attributes are: ◆ **key** A keyword naming the index. The user would be able to choose this type of search in the Resources Pane. ◆ **valuebased** Whether the index should note the presence or the value of an element or attribute. ◆ **xpath** An expression, using XPath syntax, specifying what to index. ◆ **active** Whether this index should be created.

Dynamic loading of resources and classes

exteNd Director supports *dynamic loading* of resources and classes within a resource set. Dynamic loading speeds development, because you can test changes in the resource set without having to deploy the whole project. You can also allow controlled changes in a production application by enabling dynamic loading for particular resource types, such as rules.

Dynamic loading is enabled by default when you create a new project or use the Express Portal project. When dynamic loading is enabled, resource set *vultures* watch disk locations for changes. After a specified interval, if a file has changed, the vulture fires an event with information about the changed resource item. Listeners for that resource set can examine the resource item and determine what action to take. Default listeners are programmed to flush the changed object from the resource set cache, so that the new version will be loaded when it is first requested.

📖 For more about events and listeners, see "Using events to report resource set changes" on page 79.

exteNd Director can dynamically load resources or classes or both. Settings in resourceset.xml determine what gets loaded. Each disk location has its own vulture settings.

NOTE: By default, the resource set is configured for dynamic loading. The necessary settings are described here so you will understand how dynamic loading works.

Configuring the resource set To use dynamic loading for a particular resource set, you need these settings in its **resourceset.xml** configuration file:

1 In the **settings** section, set the **vultures** element to true. This enables the resource set to look at disk locations and discover changes.

2 If you want Java classes to be dynamically loaded, set the **dynamicClassLoading** element to true. Then use the Enable/Disable button to disable resource JAR projects whose classes you want to load dynamically from disk.

3 Add one or more **entry** elements for disk locations to the **resourcePath** or **libPath** section. Each entry specifies a disk location that contains resources or classes. For each entry element, set these attributes:

- Set the **active** attribute to true.
- Set the **vultureInterval** attribute to a value greater than zero.
- Set the **recursive** attribute to true if the vulture should examine subdirectories too; false if subdirectories should be omitted.

For an example of how this looks in XML, see "Example" below.

How vultures work In resourceset.xml, if the vultures element in the settings section is set to true, and if an entry in the resourcePath or libPath section specifies a disk location whose vulture interval is greater than zero, an exteNd Director vulture watches that disk directory. When the vulture interval is reached, the vulture checks to see if any of the files within the directory have been modified. The vulture also checks for new files. Deleted files are not processed.

When the vulture finds a changed file in the disk location, it flushes the previous instance of the file, and the new version is loaded the next time the object is requested. Once an item has been loaded into the cache, it is not removed until the next time the server is started, or the next time the WAR is deployed.

Dynamic loading and the class loader The Java classloader supersedes the dynamic loading of resources and classes. So if the EAR or WAR contains a class, that class will never be dynamically loaded from disk. To load classes from disk, make sure they are not included in the deployed archive. You can use the Enable/Disable Subprojects option in the resource set configuration editor to omit one or more resource JARs from the archive.

You can enable dynamic loading for helper classes used by exteNd Director classes such as portlets. If you modify the helper classes, the classes are vultured and loaded into an instance of the class loader. However, the portlet may have been loaded by a previous instance of the class loader. In this case, it will continue to reference the classes found by its instance of the classloader. Therefore, if you recompile the helper classes, make sure to recompile the portlet as well, and you will get the latest version of the portlet and its helper classes.

Example Suppose you want to dynamically load pages, portlets, and styles from disk. The XML files are stored in a directory called MyProjects\MyEAR\MyApp\data on your C drive. The Java classes for your portlets are compiled to MyProjects\MyEAR\MyApp\build\resource-classes.

To dynamically load updated versions of resources (pages, portlet descriptors, styles), you would need to add the data directory to the resourcePath section of the resourceset.xml file. For classes, you would need to add the build directory to the libPath. You would also disable the resource JAR project so that it is omitted from the archive (necessary only if you want to dynamically load classes).

For each path entry, you would set the vulture interval attribute to indicate how often you want the vulture to check the disk locations for changes. The vulture interval is expressed in milliseconds.

These XML excerpts appear in resourceset.xml:

```
<resourceset>
<settings>
  <name>ResourceSet</name>
  <dynamicClassLoading>true</dynamicClassLoading>
  ...
  <vultures>true</vultures>
</settings>
<path-entries>
  <resourcePath>
    ...
    <entries>
      ...
      <entry active="true" vultureInterval="$FREQUENT$"
             recursive="true">$WARLOCATION$/data</entry>
    </entries>
  </resourcePath>
  <libPath>
    ...
    <entries>
      ...
      <entry active="true" vultureInterval="$FREQUENT$"
             recursive="true">$WARLOCATION$/build/resource-classes</entry>
    </entries>
```

```
  </libPath>
</path-entries>
...
</resourceset>
```

Using events to report resource set changes

As you've learned, a resource set holds its resources and classes in JARs and on disk. These locations are listed in resourceset.xml in the resourcePath and libPath sections. The contents of a resource set can change when you add or remove files from the disk locations. To recognize and act on changes, you can configure exteNd Director vultures, described in "Dynamic loading of resources and classes" on page 77.

When a vulture notes a change, it fires an UPDATE event. exteNd Director subsystems can listen for resource set events and react appropriately.

This section describes how to configure an event listener and what to do to handle the event.

NOTE: This information about resource set events is an advanced topic. The standard listeners are already configured for you. They provide the only required functionality, which is resource set caching.

Working with listeners

Listeners configured for you

In a new exteNd Director project, the Rule, Workflow, and Security subsystems are each configured with a resource set listener that is registered during the boot process. They receive events for their bound resource set.

These subsystems each have a listener class called com.sssw.<*subsystem*>.core.EboResourceListener, which extends com.sssw.fw.resource.EboResourceListener. These default listeners handle caching of classes and resources, so in most cases you will want to continue to use these listeners. You can add as many additional listener classes as you need.

Binding

Currently, a subsystem is bound to one resource set in a one-to-one relationship. A listener you register for a subsystem gets events for the one bound resource set. Changes to that resource set cause any registered listener's stateChanged() method to be called.

Adding a listener

There are two ways to add a listener for a resource set:

- Configure a listener service in a subsystem's services.xml. The listener is registered during the boot process.

  OR

- Call the addStateChangeListener() method of EboResource to register a listener.

The listener is a class that implements com.sssw.fw.util.EbiStateChangeListener.

Adding a listener during startup

To register a listener during startup, you include a service element in the services.xml file for the subsystem. For example, this XML registers the default listener for the rule subsystem:

```
<service>
  <interface>com.sssw.re.core.EboResourceListener</interface>
  <impl-class>com.sssw.re.core.EboResourceListener</impl-class>
  <description>RuleService ResourceSet Listener</description>
  <max-instances>1</max-instances>
  <startup>A</startup>
</service>
```

The impl-class element specifies the listener class that implements the methods of EbiStateChangeListener. For a listener service, the interface is typically the same as the class.

The startup element has a value of **A**, for autostart—meaning it will be registered during the boot process.

Timing issues during startup During the boot process, the target object may not be instantiated when a registration request occurs—so exteNd Director uses a delayed registration procedure that records the registration request and registers the listener after the target resource set is instantiated. When the registration occurs, a stateChanged event is fired with a status of EboState.REGISTER.

Adding or removing a listener in your code

To add a listener in your application code, you call the static method addStateChangeListener() of com.sssw.fw.resource.EboResource. The arguments are:

- Name of the resource set
- Listener class that implements com.sssw.fw.util.EbiStateChangeListener

For example, this code adds the current class as a listener for the MyResources resource set:

```
EboResource.addStateChangeListener("MyResources", this);
```

To remove a listener, call the removeStateChangeListener() method. This code removes the current class as a listener for the MyResources resource set:

```
EboResource.removeStateChangeListener("MyResources", this);
```

If the class you specify is not a registered listener, the remove request is ignored.

Types of events

Resource set events are stateChanged events, reporting changes in the status of the resource set. Status codes for the various states are defined in com.sssw.fw.util.EboState. A resource set generates two types of events:

- **EboState.REGISTERED** Reported when the event listener is registered
- **EboState.UPDATE** Reported when a change happens in one of the watched disk locations in the resource set

An EboResourceEvent object is passed to the stateChanged event. It consists of the resource element that changed and an EboState status code.

Firing an event You can also fire stateChanged events to all of a resource set's listeners. The event can use your own application-specific status codes or the EboState codes.

This sample code illustrates how you might prepare for and fire an event. The resource set is named MyResources:

```
EboResource rs = EboResource.getLoaded( name );
EboResourceElement rsrcElem = rs.findResourceElement( file );
EboResourceEvent rsrcEvt = new EboResourceEvent( rsrcElem, MYSTATUSCODE );
EboResource.fireStateChanged("MyResources", rsrcEvt );
```

What listeners do

Behavior of the standard listeners Each subsystem that uses resources has a standard listener that responds to changes in resource sets. The services.xml file for the subsystem sets up the registration of the listener.

If the resource set's vultures are turned on and a vulture notes that a change has occurred in a disk location, the vulture fires the listener's stateChanged event and passes an EboResourceEvent object containing a reference to the changed object. The listener code finds out if the changed object is relevant to that subsystem—and if so, flushes the old version of the resource from the subsystem's internal cache.

NOTE: To preserve the dynamic loading of resources, do not change the configuration of the standard listeners.

Writing your own listener Your listener class must implement the class com.sssw.fw.resource.EboResourceListener. Its only method is stateChanged() with an argument of type EboResourceEvent. The following sample code checks for this, then gets the resource element from the event object and takes some appropriate action.

The stateChanged method might look like this:

```
public void stateChanged(EboEvent eo)
{
  if (eo instanceof EboResourceEvent)
  {
    EboResourceEvent evt = (EboResourceEvent) eo;
    if (evt.getState() == EboState.REGISTERED)
    {
      ... // Code to respond to getting registered, if any
    }
    if (evt.getState() == EboState.UPDATE)
    {
      EboResourceElement elem = evt.getResourceElement();
      if (elem != null )
      {
         if (elem.getDirectoryName().startsWith("rule")
         {
                ... // Do something for a rule resource element
         }
      }
    }
  }
}
```

Validating a resource set

When instantiating a resource set, exteNd Director can run validation classes to check the resource set.

NOTE: Validation of a resource set is an advanced topic. You can enable validation in resourceset.xml and use the default validation with no further effort on your part.

The default validation checks the existence of:

- All items on resourcePath and libPath
- All portlet classes referred to in portlet descriptors

If the validate setting of the resource set is true, then during the boot process, when the resource set is instantiated, exteNd Director runs all the validation classes it finds in the resource set. They are called in no specific order.

Writing your own validation class You can write your own validation classes to check whatever needs checking in your application. These classes can be stored anywhere in the resource set JARs.

Your validation class must implement the com.sssw.fw.resource.api.EbiValidator interface. The interface contains one method:

```
public void validate( com.sssw.fw.resource.EboResource resource ) throws
Exception;
```

If a validation class throws an exception, exteNd Director will display a stack trace on the console, then proceed with the next validation class. An exception does not stop the validation process or prevent the exteNd Director application from running. All you need to do is be aware that the information on the console could indicate problems in your application. It is up to you to take further action.

Running a validation test In exteNd Director, you can run the resource set validation before you deploy:

1. In the WEB-INF/lib directory of the resource set, find and open **resourceset.xml**.
2. In the graphical view of the editor, select the **General** tab.
3. Click the **Validate** button.

   Any output from the validation classes is displayed in the editor's output window.

Storing XML files that contain MBCS characters

If you want to store XML files in a resource set that contains extended ASCII or multibyte character set (MBCS) characters, you need to add an XML header to these files that specifies the correct encoding for the locale. If the machine's locale is France, for example, the encoding should be ISO8859-1. In this case you would need to add the following header to each XML file stored in the resource set:

```
<?xml version="1.0" encoding="ISO8859_1" ?>
```

If this header is missing, the data will not be parsed correctly by the XML parser.

7 Editing the Configuration of a Resource Set

This chapter describes how to edit a project resource set. It has these sections:

About the Resource Set Editor

exteNd Director provides a custom editor for editing the configuration file for a resource set. That file is always called resourceset.xml and is in the WEB-INF/conf directory of an application WAR that uses a resource set. This section describes interesting features of the Resource Set Editor.

NOTE: The section "Configuring the resource set" on page 71 provides information about the settings you make in resourceset.xml.

➢ **To open the Resource Set Editor:**

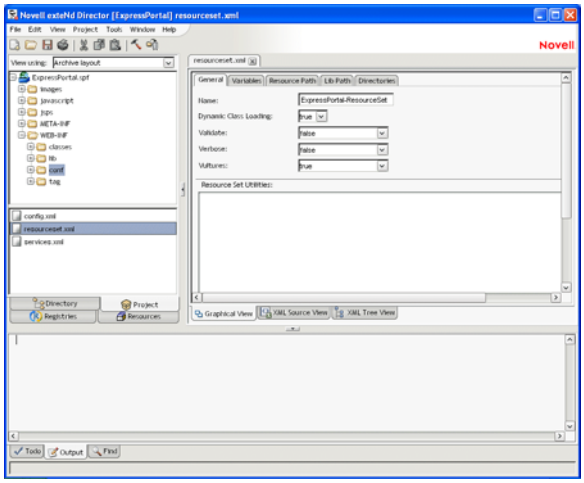
1 On the Project tab of the Navigation Pane, find **resourceset.xml** in the WEB-INF/conf directory of the WAR.

OR

On the View tab within the Resources tab of the Navigation Pane, find **resourceset.xml** in an appropriate view.

2 Double-click the file name.

The editor opens and displays one of several tabbed panes. You can edit values in this graphical view or you can select XML source view and edit the XML as text:



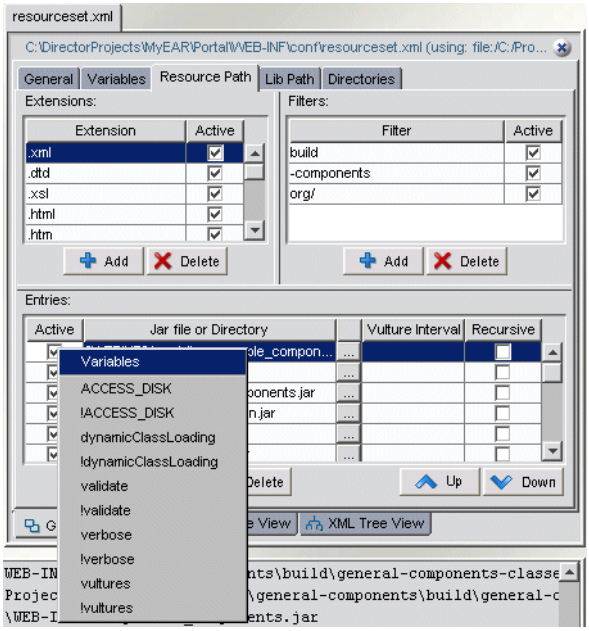
Using boolean variables in check box fields

Variables you define in resourceset.xml are available for use in any attribute or element value. In graphical view, you can choose variables from a list. The Resource Set Editor will display either a checked or an empty check box according to the current value of the variable.

➢ **To use a variable for a check box value:**

1 Right-click on or beside the check box.

A popup menu displays each boolean variable in normal and negated versions.



2 Select the variable you want from the list.

The box is checked if the variable expression is true and empty if the variable expression is false.

NOTE: To see what values use variables, switch to XML Source View and look for variables in the form **$*variablename*$** and **!$*variablename*$**.

Working with entries for resourcePath and libPath

When working with the list of entries for resourcePath and libPath, there are several points to keep in mind:

Point	Details
For check box values, such as **Enabled**, you can use a variable.	You might configure the resource set so that an entry is enabled when a variable is set to indicate the presence of a particular subsystem.
The order of entries is significant.	Use the Up and Down arrows to move entries to the correct position in the list. JARs and disk locations that you want searched first should be at the end of the list.
To specify the location of a JAR that is in the deployed WAR, specify a relative path, usually WEB-INF/lib.	You can use the variable $WEB-INF$, whose value is WEB-INF/lib.

Point	Details
To specify a disk location, specify an absolute path.	This is one of the few values that needs to be changed if you move the project to another computer. However, if you use the variable $WARLOCATION$ in the path, you will only need to redefine the variable, not all the entries, if you move the project.

Using resource set utilities

The resource set is not just a feature of your deployed application. It is also running in the development environment to enable tools like the Relationship Viewer and Search and to find resource files used by editors and wizards. To control the resource set in the development environment, the Resource Set Editor provides several utilities on the General tab.

These utilities affect the resource set in the development environment only, not the deployed application:

Button	What it does
Stop, Start, Restart	Starts or stops the resource set. Use **Restart** after changing entries on resourcePath and libPath so that the development environment can find the correct resource set contents. Use **Stop** to conserve memory when you're not editing a resource set's files or using it on the Resources tab of the Navigation Pane.
Validate	Runs the validation classes in the resource set. The default validation verifies that entries in libPath and resourcePath exist. If you've added your own validation classes, they are executed too. 📖 For information on adding custom validation classes, see "Validating a resource set" on page 81. If you set Validate to true in resourceset.xml (on the upper half of the General tab), the same validation process is run during the exteNd Director boot process on the server.
Clear	Clears the messages reported by resource set utilities.
Enable/Disable Resource Subprojects	Displays a list of resource JAR subprojects in the resource set. Clear the check boxes of projects whose JARs you want to leave out of the archive. The settings you specify will be used the next time you build the archive in the development environment. When dynamic class loading is enabled, you need to disable subprojects whose classes will be loaded from a disk location. Typically this is the *myapp*-resource subproject. To find out what is being loaded from disk, see the last entries on the libPath tab. To speed deployment, you might also disable subprojects that your application isn't using yet.

8 Using the Relationship Viewer

This chapter explains how to use the Relationship Viewer to navigate relationships within a resource set. It contains the following sections:

About the Relationship Viewer

The *Relationship Viewer* shows the relationships that exist for the currently open object. When you open an object in a resource set, the Relationship Viewer displays an easy navigation path from this object to other items within the resource set that may be of interest to you. The possible navigation paths are represented as a tree of folders and files, similar to the tree presentation used in Windows Explorer. You can expand or collapse the branches of the tree to navigate the relationships within the resource set.

The Relationship Viewer is a visual rendering of output produced by a *relationship analyzer*. The relationship analyzer determines which navigation paths are interesting to show to the user from any object within the resource set. The relationship analyzer generates an XML tree as output, which is processed by the Relationship Viewer. Typically, these navigation paths are based on relationships that are fundamental to an exteNd Director application. If you open a PID page, for example, the Relationship Viewer displays the PID descriptors for the page.

exteNd Director includes built-in relationship analyzers for the common items in a resource set. And you can create custom relationship analyzers to suit your own application requirements.

Navigating relationships within a resource set

To use the Relationship Viewer you must first open a file, and you do that from either the Search tab or a view.

➢ **To navigate the relationships for a resource:**

1. In the exteNd Director Navigation Pane, click the **Resources** tab.
2. Click the **Search** tab or the **View** tab.
3. Find the item you want to open.

    For more information on using the Search tab, see Chapter 9, “Searching a Resource Set”. For more information on the View tab, see Chapter 10, “Working with Views”.
4. Double-click the item.

   The source file editor displays the contents of the file, and the **Relationship Viewer** shows one or more relationships that are specific to the type of file you opened.

Items that were read from disk are editable and appear in black. Items that were read from JAR files are not editable. These items appear in blue and are marked with **RO** (read-only).

5 In the Relationship Viewer, expand or collapse the branches of the tree to see the possible navigation paths.

6 When you see another file you want to open, double-click it.

The source file editor displays the contents of the file, and the Relationship Viewer displays one or more relationships that are specific to this file.

Creating a custom relationship analyzer

You might create a **custom** relationship analyzer when you want to create a new type of element that an exteNd Director resource set should know about.

For example, suppose you create custom elements in a directory called **my-stuff** within the resource set. Within the my-stuff directory, you place XML files that list portlets of interest. In this case, for each XML document in the my-stuff directory, you might want to process portlet information and add each portlet listed to the XML tree.

Here's the source for example.xml, a sample file in the my-stuff directory:

```
<portlets>
  <portlet cid="FireRule.xml">
  <portlet cid="MyPages.xml">
</portlets>
```

When the example.xml file is opened in the development environment, the custom relationship analyzer is called; it reads the document and converts it into an XML structure that represents the tree that should be displayed.

The Relationship Viewer renders the tree generated by a custom relationship analyzer just as it would render any tree generated by one of the built-in relationship analyzers. Items that were read from disk are editable and appear in black. Those items that were read from JAR files are not editable. These items appear in blue and are marked with **RO** (read-only).

Creating a relationship analyzer class

To determine which relationship analyzers are available, the development environment asks the resource set for a list of all classes that implement the **EbiElementRelationshipAnalyzer** interface. When it has this list, it cycles through the classes on the list asking each if it wants to be the provider of the relationship tree. It does this by calling the **analyzeFile()** method. If the method returns true, processing is stopped and the same class is asked a second time to provide the XML structure that represents the tree of relationships. It does this by calling the **getTreeXML()** method.

To create a custom relationship analyzer, you need to create a class that implements the EbiElementRelationshipAnalyzer interface and put that class inside the resource set. For example, to support custom elements in the my-stuff directory (as described above), you might create a source file called MyStuffRelationshipAnalyzer.java and put it in the ResourceSet/src directory.

Here's the code for the MyStuffRelationshipAnalyzer:

```
import java.io.*;
import java.util.*;
import com.sssw.fw.resource.*;
import com.sssw.extensions.xwb.util.*;
import com.sssw.extensions.xwb.fw.resource.api.*;
import com.sssw.extensions.xwb.fw.resource.ui.*;
```

```
public class MyStuffRelationshipAnalyzer implements EbiElementRelationshipAnalyzer
{

public boolean analyzeFile( EboResourceElement element ) {
 // make sure we have an element
  if ( element != null ) {
   // if it is from the "my-stuff" directory & is XML, then I will
   // provide the relationship analyzer for it
   if ( element.getDirectoryName().equals( "my-stuff" ) ) {
      return element.isXML();
   }
 }
 return false;
    }

public org.jdom.Document getTreeXml( EboResourceElement element ) {
 // create the xml tree
 EboTreeXmlHelper th = new EboTreeXmlHelper();
 try {
    // add the opened object as a node
    th.addElement( th.makeElement( element,
       EbiElementRelationshipAnalyzer.XML_ICON ) );
    // create a portlets folder
    EboFolder portletsFolder = new EboFolder( "Portlets" );
    th.addFolder( portletsFolder );
    // find all the portlets nodes
    org.w3c.dom.NodeList portlets =
      element.getDocument().getElementsByTagName( "portlet" );
    if ( portlets != null ) {
        String cid = null;
        org.w3c.dom.Element portletElem = null;
        int len = portlets.getLength();
        // process through the portlets nodeList
        for ( int i = 0; i < len; i++ ) {
            if ( portlets.item( i ) instanceof org.w3c.dom.Element ) {
            portletElem = (org.w3c.dom.Element)( portlets.item( i ) );
            // get the portlet id ( cid )
            cid = portletElem.getAttribute( "cid" );
            if ( cid != null ) {
             // try and find the portlet in the portal-portlet
             // directory
                EboResourceElement re =
                EboResource.getCurrent().getResourceElement(
                    "portal-portlet", cid );
                // add it to the portlets folder
                th.addElement( portletsFolder, th.makeElement( re,
                    EbiElementRelationshipAnalyzer.XML_ICON ) );
                        }
                    }
                }
            }
        }
        catch( Exception e ) {
            System.out.println( e );
        }
        return th.getDocument();
    }

}
```

9 Searching a Resource Set

This chapter explains how to search the contents of a resource set. It contains the following sections:

About the Search tab

The **Search tab** provides a facility for searching for objects within a resource set. The Search tab lets you specify search filters for locating objects. You can search by file name or directory name, or look for particular string patterns within files in a resource set. When you've found the objects you're looking for, you can make any changes you want, or save the search as a view.

The Search tab provides complete support for **regular expression** searches.

For syntax and reference information on regular expressions, see the section on regular expressions for text searches in *Utility Tools*.

Searching a resource set

➢ **To search a resource set:**

1 In the exteNd Director Navigation Pane, click the **Resources** tab.

2 Click the **Search** tab.

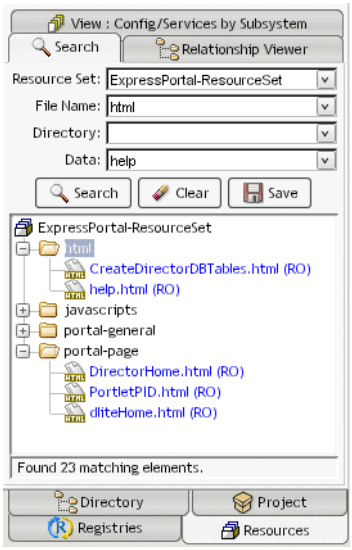
3 Specify the following:

Field	What to specify
Resource Set	The target resource set for the search operation.
File Name	A regular expression that specifies the list of file names to search for.
Directory	A regular expression that specifies which directories should be included in the search. If you want to search only a single directory within the resource set, select from the dropdown list.
Data	A regular expression that specifies a string pattern to search for within files in the target resource set.

If you specify values for multiple fields, the field values are combined together in an AND operation.

4 Click **Search**.

The Search tab displays the list of files that match the search filter, organized by folder. For example, the following search locates all **HTML** source files that contain the string **help**:



Items that were read from disk are editable and appear in black. Items that were read from JAR files are not editable. These items appear in blue and are marked with **RO** (read-only).

➢ **To clear the search criteria:**

- In the **Search** tab, click the **Clear** button.

➢ **To open an item displayed in the Search tab:**

- Double-click the item in the Search tab.

  The source file editor displays the contents of the file, and the **Relationship Viewer** shows one or more relationships that are specific to the type of file you opened.

  📖 For more information on using the Relationship Viewer, see Chapter 8, “Using the Relationship Viewer”.

Saving a search as a view

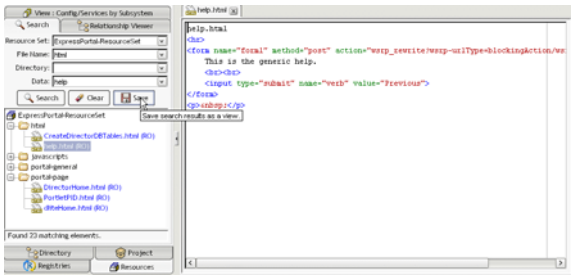
If you execute the same search operation frequently, you may want to consider saving the search as a view. That way you can execute the search as often as you like without having to specify the search criteria.

When you save a search as a view, the Search tab creates a view definition file in the **my-views** folder within the resource set. The view definition file is an XML file that specifies which items should be included in the view.

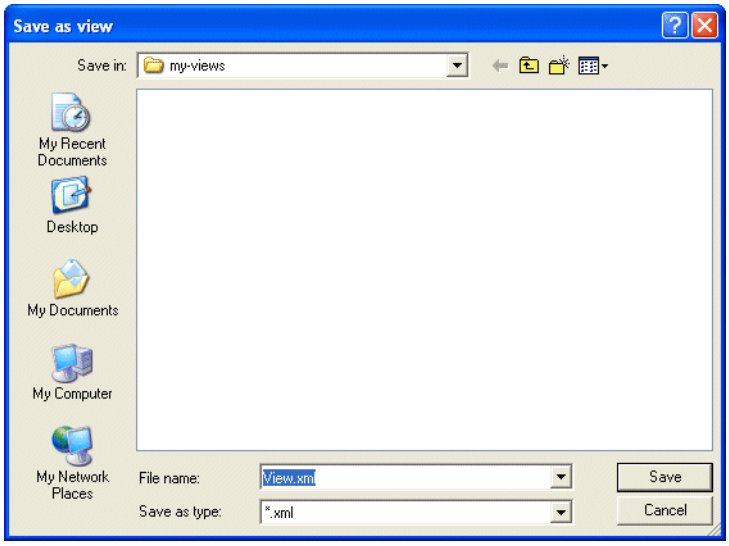
Creating a view manually You can also create a view by manually editing a view definition file. For more information on defining views manually, see "Defining custom views" on page 106.

➢ **To save a search as a view:**

1 In the **Search** tab, click **Save**.



The Save dialog displays:



2 Specify a name for the view definition XML file.

3 Click **Save**.

Working with the Search API

The Search tab uses the underlying Search API provided with exteNd Director. For many applications, this search facility is satisfactory. But sometimes you may also want to search the contents of a resource set programmatically. To do this you need to use the following objects in the Search API:

Object	Description	Implementation class(es)
Search object	Performs the search operation. Returns a Vector of resource elements (EboResourcElement objects) that match the search criteria.	EboResourceSearch
Search template	Defines the pattern or set of characteristics that will be used to filter the search.	EboResourceElementTemplate
Search comparator	Compares the template to the target object in the resource element.	◆ EboResourceElementComparator ◆ EboResourceClassElement Comparator ◆ EboResourceREGEXPElement Comparator

All of the search implementation classes are located in the **com.sssw.fw.resource.search** package.

How it works The *search object* performs the search operation by iterating through the target objects in the resource set, comparing each object with the search template. The *search template* defines the pattern or characteristic that will be used to filter the search. For example, the search template can specify a file name, directory name, disk path, or string of bytes to search for. When the search is performed, the template is compared against each target object in the resource element. If the objects are equivalent, the target object is added to the search results. The search object returns the results in a Vector of resource element objects.

Ways to set the search template There are several ways to set the search template object for a search:

- Use the internal search template object
- Create your own search template object and pass it to the search object
- Pass a DOM object directly to the search object

The **search comparator** compares the template against the target object to see if there's a match. The Search API includes three search comparators:

Comparator	Description
EboResourceElement Comparator	Supports comparison using the indexOf() method on the String class.
EboResourceClassElement Comparator	Supports comparison using the indexOf() method on the String class. This comparator filters out items that end in .class.
EboResourceREGEXPElement Comparator	Supports comparison using regular expressions.

Example 1: using the internal search template object

Suppose you want to search for all XML files in a resource set that contain the string FireRule. One way to do this would be to use the internal template object built into the EboResourceSearch class. Here's a code snippet that shows how you might do this:

```
EboResourceSearch resourceSetSearch = new EboResourceSearch();
resourceSetSearch.setComparator(
    new EboResourceREGEXPElementComparator() );
resourceSetSearch.getTemplate().setFileName( ".xml" );
resourceSetSearch.getTemplate().setBytes( "FireRule" );
Vector v = resourceSetSearch.search( resourceSet );
```

This code example uses EboResourceREGEXPElementComparator as the search comparator. This is the search comparator for regular expression comparisons.

Example 2: creating your own search template object

You can also define your own search template object and instruct the search object to use that instead of the built-in search template object. In this case you instantiate the template object, fill in the template values you want to use, and then pass the object to the search object. The following example shows how you might do this:

```
EboResourceSearch resourceSetSearch = new EboResourceSearch();
resourceSetSearch.setComparator(
    new EboResourceREGEXPElementComparator() );
EboResourceElementTemplate template = new EboResourceElementTemplate();
template.setFileName( ".xml" );
template.setBytes( "FireRule" );
resourceSetSearch.setTemplate( template );
Vector v = resourceSetSearch.search( resourceSet );
```

Example 3: serializing a search request

If you want to be able to save a search as a view, you need to serialize the search request. To do this you need to build XML that conforms to the following DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT search EMPTY>
<!ATTLIST search
  bytes CDATA #IMPLIED
  directoryName CDATA #IMPLIED
  diskPath CDATA #IMPLIED
  javaPackage CDATA #IMPLIED
  fileName CDATA #IMPLIED >
```

Suppose you want to search for all XML files in a resource set that contain the string FireRule. The XML string required for this search would be:

```
<search fileName=".xml" bytes="FireRule" />
```

To build this XML string programmatically, you could use the following code:

```
EboResourceSearch resourceSetSearch = new EboResourceSearch();
resourceSetSearch.setComparator(
   new EboResourceREGEXPElementComparator() );
resourceSetSearch.setTemplate( "<search fileName=\".xml\" bytes=\"FireRule\" />"
);
Vector v = resourceSetSearch.search( resourceSet );
```

10 Working with Views

This chapter explains how to use views to look at exteNd Director project items. It contains the following sections:

About views

You can use *views* to display personalized lists of items within an exteNd Director project. When you display a view, you see the list of items in the view represented as a tree of folders and files, similar to the tree presentation used in Windows Explorer. You can expand or collapse the branches of the tree to locate the items you want to edit.

Views can be used to look at resources in a resource set, or at system configuration and service settings. exteNd Director ships with several predefined views and also allows you to define custom views to display project items that are of particular interest to you.

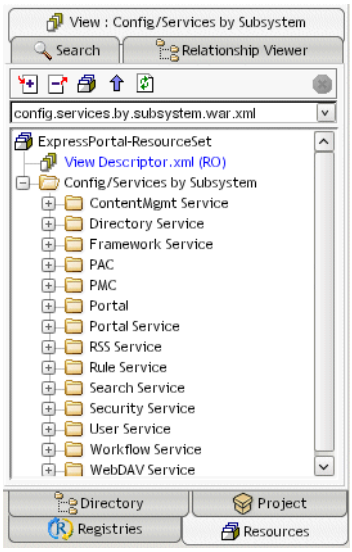
View definitions are stored as XML documents in the **my-views** folder within a resource set. The predefined views that ship with exteNd Director are provided in the **analyzers_views.jar**, which is added to the WEB-INF\lib directory within the WAR for your application.

Displaying a view

➢ **To display a view:**

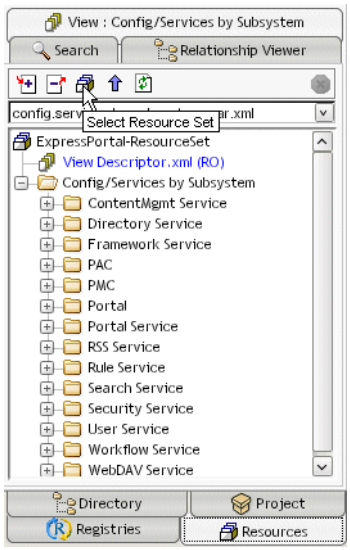
1. In the exteNd Director Navigation Pane, click the **Resources** tab.
2. Select the **View** tab, if it is not already selected.

The View tab displays the default view for the current resource set. In a WAR project, the default view is **config.services.by.subsystem.war.xml**. In an EAR project, the default view is **config.services.by.subsystem.ear.xml**. This view displays the config.xml and services.xml files for each subsystem:



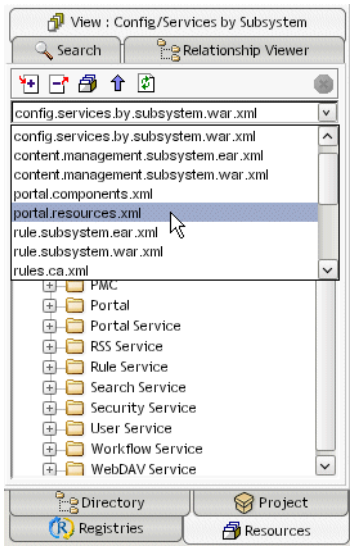
3 If you have more than one resource set:

3a Click the **Select Resource Set** button:

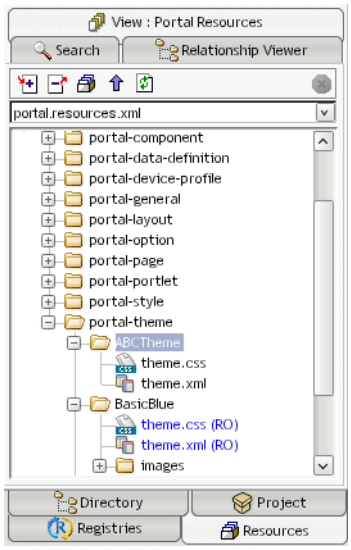


3b Select the target resource set and click **OK**:

4 Select the view you want to see in the view dropdown box:

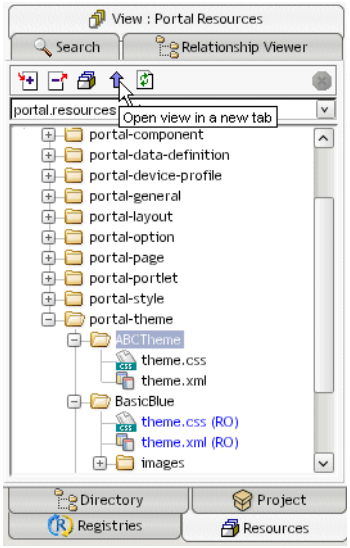


Items that were read from disk are editable and appear in black. Items that were read from JAR files are not editable. These items appear in blue and are marked with **RO** (read-only).



➢ **To open a view in a separate tab:**

- Click the **Open view in a new tab** button in the view display window:



➢ **To open an item displayed in a view:**

- Double-click the item in the view display window.

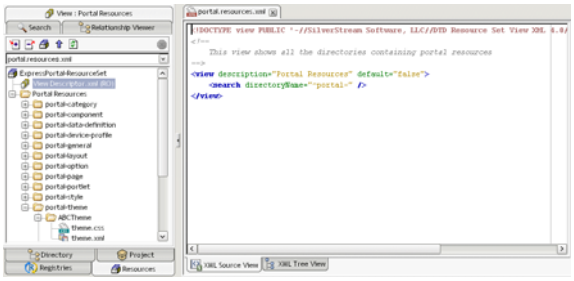
  The source file editor displays the contents of the file, and the **Relationship Viewer** refreshes to show one or more relationships that are specific to the type of file you opened.

  To see these relationships, click the **Relationship Viewer** tab.

  📖 For more information on using the Relationship Viewer, see Chapter 8, “Using the Relationship Viewer”.

➢ **To see the definition of a view:**

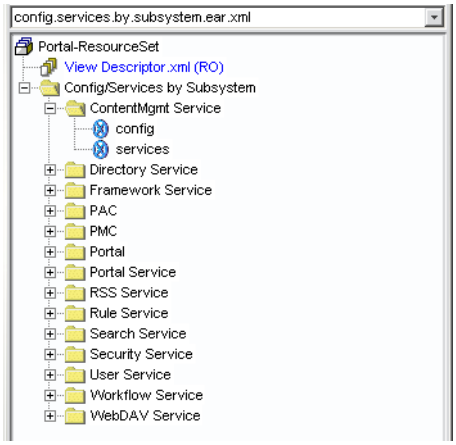
- Click **View Descriptor.xml** under the root node in the view display window:

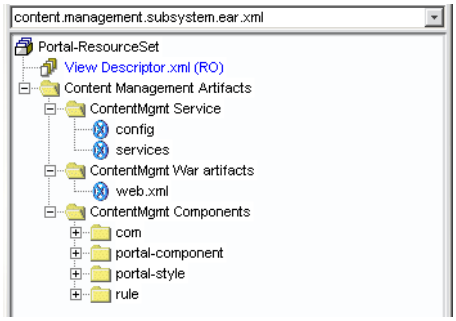


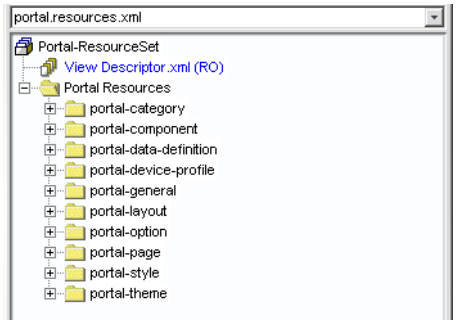
Using predefined views

exteNd Director provides a set of predefined views that you may find useful for locating project files. This section describes each predefined view in detail. You can use these views as provided or copy and modify them to suit your requirements.

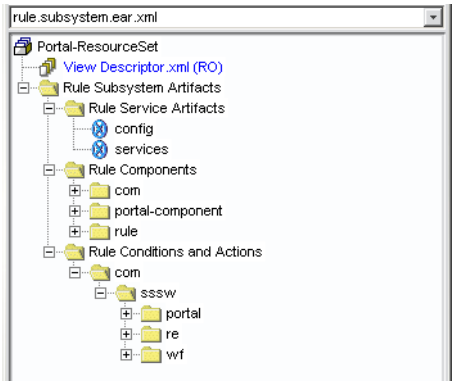
View	What it shows

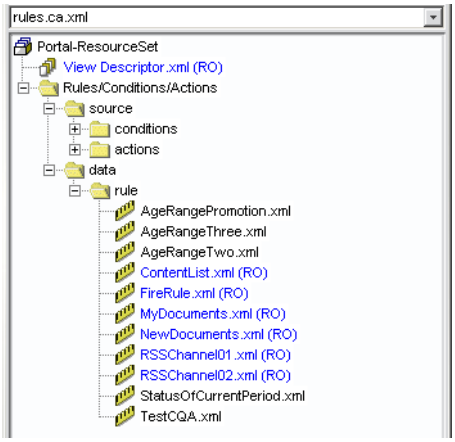
config.services.by.subsystem.ear.xml
config.services.by.subsystem.war.xml
Settings for all exteNd Director subsystems.
Searches for:
◆ Configuration files:
`config.xml`
`services.xml`
	

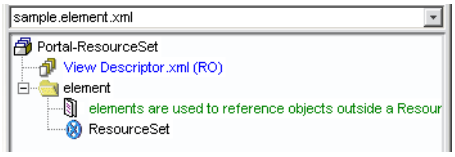
content.management.subsystem.ear.xml
content.management.subsystem.war.xml
Artifacts related to the Content Management subsystem.
Searches for:
◆ Configuration files:
`config.xml`
`services.xml`
`web.xml`
◆ Files whose names contain:
`Content`
`CM`
◆ In these directories:
`com`
`portal-component`
`portal-style`
`rule`
	

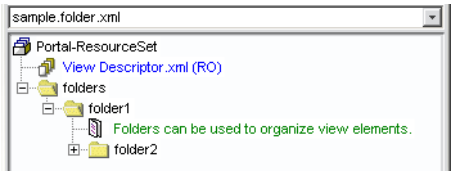
portal.resources.xml All directories containing portal resources Searches for: ◆ Directories whose names begin with: `portal-`	

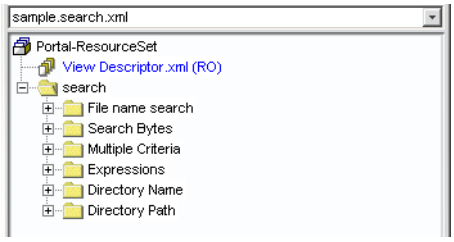
View	What it shows

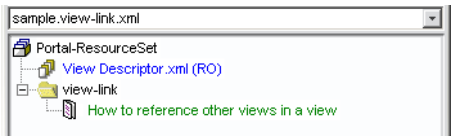
rule.subsystem.ear.xml
rule.subsystem.war.xml
All artifacts related to the Rules subsystem
Searches for:
◆ Configuration files:
`config.xml`
`services.xml`
◆ Files whose names contain the strings:
`rule`
`Rule`
◆ In these directories
`portal-component`
`portal-styles`
`rule`
◆ Files whose names contain:
`.class`
`.java`
◆ And whose contents contain the string:
`portal.condition`
`portal.action`
`re.condition`
`re.action`
	

rules.ca.xml
All rules and conditions and actions
Searches for:
◆ Java source files whose content contains the regular expression:
`implements.*EbiCondition`
`extends.*Condition`
`implements.*EbiAction`
`extends.*Action`
◆ Directories whose names begin with:
`rule`
	

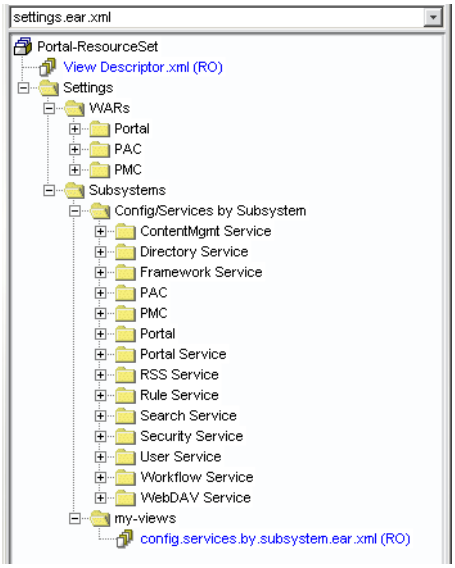
sample.element.xml
How to use a element in a view
Elements are used to reference objects outside the scope of a resource set
Searches for:
◆ Files whose names contain:
`ResourceSet`
◆ In the disk path:
`$WARLOCATION$/web-inf/conf/resourceset.xml`
	

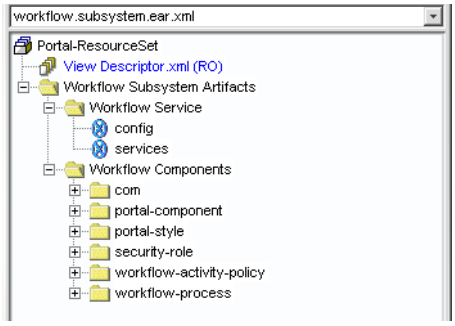
View	What it shows
sample.folder.xml How to use a folder in a view Searches for: ◆ Nothing	

sample.search.xml
How to use a search in a view
Searches for:
◆ Files whose names contain:
`my`
◆ Files whose contents contain:
`Weather`
◆ Files whose names contain:
`Phone`
◆ And whose contents contain:
`com/sssw/portal/component`
◆ Files whose names contain:
`Stock`
`Weather`
◆ And whose contents contain:
`com/sssw/portal/component`
◆ Directories whose names contain:
`security`
◆ Files within the disk path:
`c:\projects\director`
	

sample.view-link.xml
How to use a view-link in a view
Searches for:
◆ The folder:
`samples`
◆ In the view definition:
`element.xml`
◆ The folder:
`samples`
◆ In the view definition:
`folder.xml`
	

View	What it shows

settings.ear.xml
settings.war.xml
Common settings for a director EAR or WAR
Searches for:
◆ Files whose names contain:
`resourceset.xml`
`web.xml`
`config.xml`
`services.xml`
◆ In the disk path:
`$WARLOCATION$/web-inf/...`
`$EARLOCATION$/PAC/web-inf/...`
`$EARLOCATION$/PMC/web-inf/...`
	

workflow.subsystem.ear.xml
workflow.subsystem.war.xml
All artifacts related to the Workflow subsystem.
Searches for:
◆ Configuration files:
`config.xml`
`services.xml`
◆ Directories whose names begin with:
`workflow`
◆ Files whose names contain:
`Workflow`
◆ In directories whose names contain:
`portal-component`
`portal-style`
`security-role`
◆ Files whose names contain:
`Workflow`
◆ And whose contents contain:
`com/sssw/portal/component`
	

Importing resources into a view

You can import the contents of a JAR into a view. When you do this, exteNd Director:

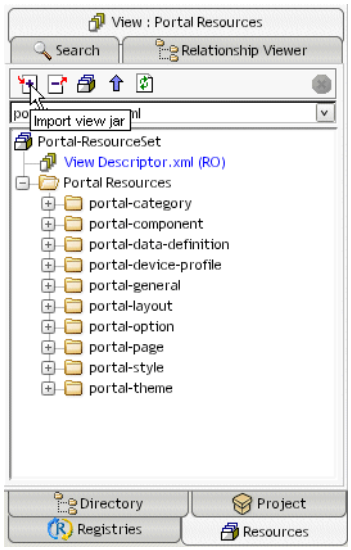
- Copies the JAR to the WEB-INF\lib directory within the resource set WAR.
- Adds an entry for the JAR to the resource path and lib path settings for the resource set.
- Adds an entry for the JAR to the project settings in the development environment.
- Restarts the resource set so that the imported resources are immediately available for use within the development environment.
- If the JAR selected for import contains a view definition XML file, creates a new view in the target resource set. This view has the same definition as the view from which the resources were originally exported.
- Displays the new view if one was created.

➢ **To import resources into a view:**

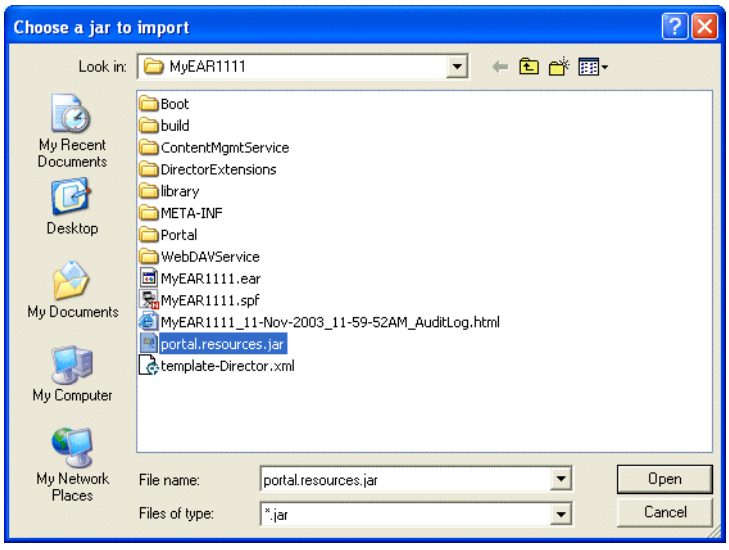
1 In the Navigation Pane, click the **Resources** tab.

2 Select the view you want to be the target for the import.

3 Click the **Import view jar** button in the view display window:



4 Select the JAR you want to import and click **Open**:



Exporting resources from a view

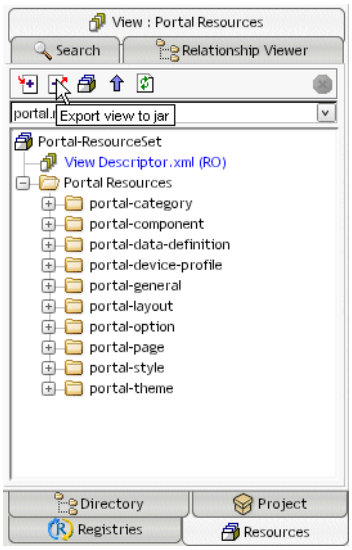
exteNd Director allows you to export the contents of a view to a JAR. When you export resources from a view, exteNd Director creates a JAR that contains all of the elements in the view, including the directory structure—plus the XML file that defines the view from which the resources were exported. This JAR can then be imported into another resource set.

➢ **To export resources from a view:**

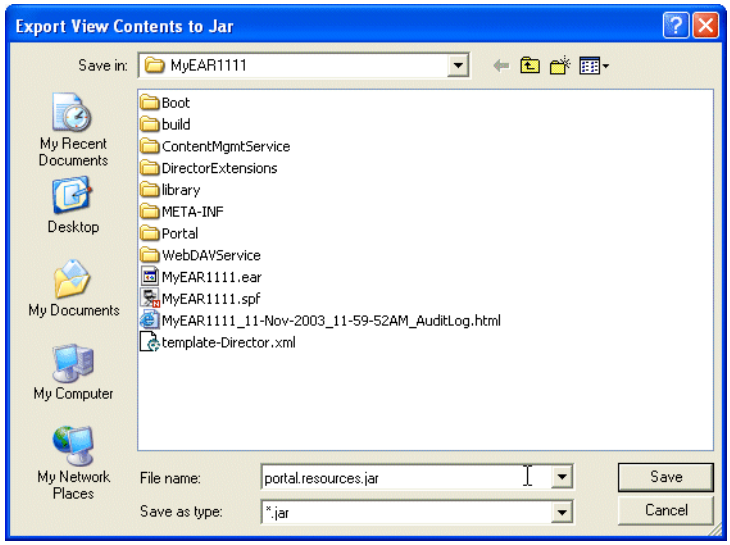
1 In the Navigation Pane, click the **Resources** tab.

2 Select the view from which you want to export resources.

3 Click the **Export view to jar** button in the view display window:



4 Specify the target directory and click **Save**:



Defining custom views

You can create custom views in one of two ways:

- Manually editing a view definition file
- Saving a search

This section describes how to manually edit a view definition file.

For details on saving a search as a view, see "Saving a search as a view" on page 93.

About the view definition file

The view definition file is an XML file that specifies which items should be included in the view. The XML for a view definition must conform to the rules specified in the **resourceset-view_4_0.dtd** file, which you can find in the DTD folder within the FrameworkService.jar.

Every view definition has a **view** element that provides a description for the view and indicates whether this is the default view:

```
<view description="myview" default="false">
    ...
</view>
```

The view description is used as the display text for the view folder.

The view element can contain one or more other elements that specify which items should be included in the view. Many of the elements support the use of **regular expression** searches.

For syntax and reference information on regular expressions, see the section on regular expressions for text searches in *Utility Tools*.

Searching for items within a resource set

You can search for items within a resource set by including one or more **search** elements in your view definition file. You can use the attributes of the search element to search for:

- File names
- Bytes within files
- Directory names
- Disk paths
- Java packages

For example, to search for all files that have the string My in their names, you would include the following search element within the view element:

```
<view description="mysearch" default="false">
    <search fileName="My" />
</view>
```

This view would show a folder called mysearch that contains all subfolders within the resource set that have files that contain My in their names.

To restrict this search to include those files that are located in directories that begin with either the string portal or the string rule, you could specify this search:

```
<search fileName="My" directoryName="^portal|^rule" />
```

If you specify multiple attributes in the search element, the attributes you specify are combined together in an AND operation.

Defining folders in a view

You can define custom folders within a view by using the **folder** element. This gives you a way to organize resources or other items of interest any way you like. For example, you might define folders in a view to categorize the results of two separate searches:

```
<view description="mysearch" default="false">
 <folder description="MyPortalItems">
  <search fileName="My" directoryName="^portal" />
 </folder>
 <folder description="MyOtherItems">
  <search fileName="My" directoryName="^rule" />
 </folder>
</view>
```

Including elements that are outside a view's resource set

A view can display items that are outside the scope of a resource set, by using the **element** tag. The element lets you specify a disk path and a filter for retrieving files by name.

For example, you could use the element tag to display various system configuration files. Here's a view definition that displays the configuration and services files for each subsystem in an exteNd Director EAR project:

```
<view description="Config/Services by Subsystem" default="false">
 <folder description="ContentMgmt Service">
  <element fileName="config"
   diskPath="$EARLOCATION$/library/ContentMgmtService/
   ContentMgmtService-conf/config.xml" />
  <element fileName="services"
   diskPath="$EARLOCATION$/library/ContentMgmtService/
   ContentMgmtService-conf/services.xml" />
 </folder>
 <folder description="Directory Service">
  <element fileName="config"
   diskPath="$EARLOCATION$/library/DirectoryService/
   DirectoryService-conf/config.xml" />
  <element fileName="services"
   diskPath="$EARLOCATION$/library/DirectoryService/
   DirectoryService-conf/services.xml" />
 </folder>
 <folder description="Framework Service">
  <element fileName="config"
   diskPath="$EARLOCATION$/library/frameworkservice/
   frameworkservice-conf/config.xml" />
  <element fileName="services"
   diskPath="$EARLOCATION$/library/frameworkservice/
   frameworkservice-conf/services.xml" />
 </folder>
...
</view>
```

Referencing other views within a view definition

A view definition can include output generated by another view. This simplifies the creation of complex views by allowing you to reuse view definitions already created. To reference another view, you need to use the **view-link** element.

The following view definition includes output from three other views:

```
<view description="view-link" default="false">
    <comment description="How to reference other views in a view" />
    <view-link folder="samples" view="comment.xml" />
```

```
    <view-link folder="samples" view="element.xml" />
    <view-link folder="samples" view="folder.xml" />
</view>
```

IV Working with Core Technologies

Guidelines for many of the fundamental technologies that can be used in building exteNd Director applications.

11 Coding Java for exteNd Director Applications

This chapter gives an overview of how to access exteNd Director services programmatically. It contains the following sections:

About coding Java for exteNd Director applications

To write Java code for exteNd Director applications, you use exteNd Director API classes in your Java code and call their methods. The exteNd Director API provides public classes (and interfaces) organized into several packages, which themselves are organized by subsystem.

The exteNd Director API is based on the Java 2 APIs (J2SE and J2EE). That means it includes classes that inherit from Java 2 classes and implement Java 2 interfaces. If you're familiar with the Java 2 APIs, you'll have a good foundation for understanding and using the exteNd Director API.

Using Java

Java is a standard language for Web applications and you'll use it in a standard way when developing exteNd Director applications. For instance, you'll:

- Write **classes** that represent the objects in your application, including **fields** (variables) for each object's data and **methods** for the actions it can perform
- Organize classes in **packages**, directory-like hierarchies that let you group related classes and make them easy to locate
- Bundle packages and classes in **JARs** (and other archive files), used for providing smaller, faster downloads to clients and for facilitating certain deployment operations

Java platform support

exteNd Director supports the **Java 2 platform**, including:

- Java 2 Platform, Standard Edition (**J2SE**)
- Java 2 Platform, Enterprise Edition (**J2EE**)

These encompass the core Java language and a variety of Java APIs.

About the core language

The core Java language is the syntax you use to perform basic programming chores. It includes:

- Comments
- Variable declarations and primitive data types
- Operators and expressions
- Statements for:
  - Defining classes, interfaces, and methods
  - Creating object instances from classes and calling their methods
  - Looping and branching
  - Handling exceptions (errors)

Much of this syntax is modeled after C and C++. JavaScript programmers will find some similarities too (although Java and JavaScript differ in other significant ways).

About APIs

API stands for *application programming interface*. In Java, an API is a collection of public classes (in one or more packages) that:

- Offers a **reusable solution** for implementing a particular area of application features or services
- Defines **public methods** you can call (and possibly public fields you can access)
- Provides a **published specification** (typically in javadoc format)

For business programming, where productivity is especially important, you'll always access one or more APIs. For example, the Java standard (J2SE) API provides many of the most fundamental capabilities you'd want to build into any application (including support for graphical user interfaces, input/output, data type manipulation, threading, networking, security, SQL, internationalization, and a lot more). There's no need to develop these capabilities yourself.

Other Java and vendor APIs (such as the exteNd Director API) take you beyond generic application services to fulfill higher-level system and business needs.

Resources for learning Java

If you're new to Java or just need to explore a specific Java topic, try the following recommended learning resources.

Books There are many other Java books available, but some good ones are:

- *Core Java* by Cay S. Horstmann and Gary Cornell, published by Prentice-Hall
- *Java in a Nutshell* by David Flanagan, published by O'Reilly & Associates
- *Teach Yourself Java 2 in 21 Days* by Laura Lemay, published by Sams
- *The Java Programming Language* by Ken Arnold and James Gosling, published by Addison-Wesley

Web sites There are many Java sites on the Web, but some good ones are:

- java.sun.com
- www.gamelan.com
- www.sys-con.com/java
- www.javaworld.com

Using the Java APIs

When building an application, you'll use particular Java APIs depending on the features or services that application requires. To help you choose which APIs you need, Sun has grouped them in different editions of the Java 2 platform:

Edition	Description
J2SE includes the standard API	Serves as the foundation for virtually any Java application you build
J2EE includes several APIs	For adding specific enterprise-level features and services to a Java application, including: ◆ Enterprise JavaBeans (EJB) ◆ Java Servlets ◆ JavaServer Pages (JSP) ◆ Java Portlets ◆ JDBC Standard Extension ◆ Java Transaction (JTA) ◆ JavaMail ◆ Java Message Service (JMS) ◆ Java Naming and Directory Interface (JNDI) ◆ RMI-IIOP ◆ JavaBeans Activation Framework (JAF)

Resources for learning J2EE

Once you're familiar with the basics of Java (including the core language and J2SE API), you can learn about J2EE and its use from the following resources.

Resource	Description
Java 2 SDK, Enterprise Edition Documentation Bundle	An index to J2EE learning and reference materials from Sun, with links to various documents and Web sites
API specification	A reference guide to the J2EE APIs, in javadoc format

Both of these resources are available at java.sun.com.

Using the exteNd Director API

- The exteNd Director API provides public classes and interfaces organized into several packages, which themselves are organized by subsystem.

exteNd Director API packages

Use the following table to find the packages that provide the major exteNd Director features or services you want in your application:

Functional area	Packages
Content Management	com.sssw.cm.api
	com.sssw.cm.client
	com.sssw.cm.event.api
	com.sssw.cm.event.util
	com.sssw.cm.factory
	com.sssw.cm.task.api
	com.sssw.cm.util
Directory	com.sssw.fw.directory.api
	com.sssw.fw.directory.client
Framework	com.sssw.fw.api
	com.sssw.fw.cachemgr.api
	com.sssw.fw.event.api
	com.sssw.fw.event.factory
	com.sssw.fw.exception
	com.sssw.fw.factory
	com.sssw.fw.log
	com.sssw.fw.resource
	com.sssw.fw.resource.api
	com.sssw.fw.resource.factory
	com.sssw.fw.resource.search
	com.sssw.fw.task.api
	com.sssw.fw.task.event
	com.sssw.fw.task.factory
	com.sssw.fw.timer

Functional area	Packages
Portal	com.novell.afw.portal.proxy
	com.novell.afw.component.api
	com.novell.afw.component.factory
	com.novell.afw.portlet.api
	com.novell.afw.portlet.consumer.factory
	com.novell.afw.portlet.factory
	com.sssw.portal.api
	com.sssw.portal.factory
	com.sssw.portal.util
Rules	com.sssw.re.api
	com.sssw.re.core
	com.sssw.re.exception
	com.sssw.re.factory
Search	com.sssw.search.api
	com.sssw.search.client
	com.sssw.search.factory
Security	com.sssw.fw.security.api
	com.sssw.fw.security.client
User	com.sssw.fw.usermgr.api
	com.sssw.fw.usermgr.client
Utilities and Helper classes	com.novell.afw.util
	com.sssw.fw.util
	com.sssw.fw.util.crypto
	com.sssw.fw.util.http
	com.sssw.fw.util.jndi
WebDAV	com.sssw.webdav.client
	com.sssw.webdav.common
	com.sssw.webdav.event.api
Workflow	com.sssw.wf.activity
	com.sssw.wf.api
	com.sssw.wf.client
	com.sssw.wf.exception
	com.sssw.wf.factory
	com.sssw.wf.link
	com.sssw.wf.ui.api

exteNd Director API terminology

The name **exteNd Director API** refers to all of the public packages. You'll also see the term **API** applied to certain subsets of these packages. For instance, the name **Content Management API** is typically used to refer to this group of packages:

- com.sssw.cm.api
- com.sssw.cm.client
- com.sssw.cm.event.api
- com.sssw.cm.event.util
- com.sssw.cm.factory
- com.sssw.cm.task.api
- com.sssw.cm.util

Just remember that these other APIs are simply convenient labels for talking about specific portions of the full API.

exteNd Director API reference documentation

exteNd Director provides a complete API specification in javadoc format. This specification details all of the packages, classes, interfaces, and members in the public exteNd Director API and includes links into the Java 2 API documentation. It's an indispensable reference for all the exteNd Director programming you do in Java.

 For complete reference information on the exteNd Director API, see the *API Reference* book in online help.

Accessing subsystem services

To access the services of an exteNd Director subsystem, you first need to use a factory to get a reference to a manager object for the subsystem. You can do this in one of the following ways:

- Get a **delegate** for a subsystem service
- Get a direct reference to a **manager object** for the subsystem

When you use a delegate, you do not need to know or care whether the service is using a local manager object or a remote object. Therefore, in most situations, you should use delegates to access subsystem services.

Each subsystem provides one or more factory classes called EboFactory that are suitable for accessing manager objects for the subsystem.

Once you have a reference to the manager object, you can call methods on that object just as you would call methods on any Java class.

Accessing a subsystem service by using a delegate

Several exteNd Director subsystems let you use delegates to access subsystem services. A *delegate* is a wrapper that hides the location of a service. The delegate model follows the J2EE Business Delegate pattern.

When you use a delegate, you do not need to know or care whether the service is using a local manager object or a remote object. The delegate initially attempts to instantiate a local manager. If this fails, it attempts to use the remote object instead. This approach allows developers to use the same code on clients and servers to instantiate services.

exteNd Director provides one or more delegates per manager. For example: the Content Management subsystem has a single delegate, but the User subsystem has four delegates (User, Group, UserMeta, and UserPersonalization).

To use a delegate to access a subsystem service, you need to call a delegate accessor method on the custom EboFactory class for the subsystem you want to use. The EboFactory class that has the method you need is typically located in the subsystem package hierarchy in a subpackage called **client**.

The delegate model is supported by the following subsystems:

- Content Management
- Directory
- Security
- Search
- User Management
- Workflow

Examples For example, to use the Content Management delegate to access content management services, you might execute this code:

```
import com.sssw.cm.api.*;
...
EbiContentMgmtDelegate contentMgr =
com.sssw.cm.client.EboFactory.getDefaultContentMgmtDelegate();
...
```

Once you have a reference to the delegate, you can simply invoke methods on the delegate. Here's an example that shows how you might do this:

```
EbiDocument tempdoc = contentMgr.getDocument(context,selectedDoc);
```

Similarly, to use the User delegate to access User subsystem services, you might execute this code:

```
import com.sssw.fw.usermgr.api.*;
...
EbiUserDelegate userMgr = com.sssw.fw.usermgr.client.EboFactory.getUserDelegate();
EbiUserInfo userinfo = userMgr.createUser(context);
...
```

Using a delegate to access a local manager The delegate initially attempts to instantiate a local manager. If this fails, it attempts to use a remote object instead. This approach can mask errors on an attempt to use a local manager only. To help you identify situations where the local instantiation fails, the delegate constructors display an informational message in the log when an instantiation fails.

You can also force the delegate to use a local manager by using a parameterized delegate factory constructor. To do this, you pass in a string indicating that you want a local delegate only. The EbiDelegate interface provides a constant you can use to indicate that you want a local delegate. If the constructor fails, it will not swallow the instantiation error on the local manager. Here's an example:

```
import com.sssw.fw.api.*;
...
EbiUserDelegate ud =
   com.sssw.fw.usermgr.client.EboFactory.getUserDelegate(EbiDelegate.SERVICE_LOCAL
);
```

Getting a direct reference to a subsystem manager

Some of the exteNd Director subsystems provide a way to get a manager object directly. For example, the EboFactory class for the Portal subsystem (com.sssw.portal.factory.EboFactory) has several methods you can use to get manager objects, such as:

- getOptionManager()
- getPageManager()
- getPortalManager()
- getPresentationManager()
- getStyleManager()
- getThemeManager()

For example, to get a reference to a portal manager from a portlet you might use this code:

```
import javax.portlet.*;
...
PortletContext context = getPortletContext();
EbiPortalManager portalMgr =
  com.sssw.portal.factory.EboFactory.getPortalManager(context);
String userUUID =
   portalMgr.getUserPortalInfo(context).getUserUUID();
...
```

Some of the subsystems for which delegates are provided also support direct access to manager objects. This support can be used in situations where the subsystem is running locally.

Accessing a rule manager The Rule subsystem requires that you use a different technique to get a manager. The Rule subsystem allows you to work with multiple rule manager instances—unlike the other subsystems, which only allow you to have a single manager instance. Therefore, to use a rule manager, you need to instantiate the object, as shown here:

```
EbiRuleManager rm =
  com.sssw.re.factory.EboFactory.createRuleManager( "sample" );
```

Handling exceptions

Errors and exceptions can occur in an exteNd Director application and can come from many sources: for example, bad input from a user, problems with the application server, database errors, and inappropriate operations on exteNd Director objects.

In developing your application, you need to plan how to handle errors and exceptions and how to let the user know the application's state. Your team will want to establish guidelines for consistently displaying error messages. Most applications will want to fulfill a set of error-handling goals such as these:

Goal	Information
Handle system exceptions	For example, when users try to access objects for which they don't have permission, you might inform them or redirect them to appropriate areas of the application. When an error occurs on the server or database, let the user know what to expect next.
Handle application exceptions	For example, your application needs to handle attempts to process an object in a way exteNd Director didn't intend or bad argument values that come from bad user input or saved data.
Provide feedback to users	A user needs to know whether requested operations succeeded or failed. If something fails because of the user's bad input, you need to supply a specific message telling the user what to do to make the input right.
Leave objects and data in a consistent state after an exception occurs	Treat operations on your business data transactionally. Plan for rolling back partially done operations so that the system correctly reflects what the user thinks happened. For example, if the user cancels an order or a system error occurs after you've saved most of the order data, be prepared to correct the state of the whole order.

Java provides the opportunity to handle errors in a reliable, consistent way. However, you must be aware of how the application flow is affected by the way you catch exceptions. For practical advice, see *Practical Java Programming Language Guide* by Peter Haggar, published by Addison-Wesley.

The error handling discussed here is an extension of standard Java error handling that you should do in any Java application. For more information, see the Java documentation from Sun Microsystems.

Errors thrown by the exteNd Director API

The following table describes the base exception classes in the com.sssw.fw.exception package. An exception can occur for various reasons during the runtime processing of an exteNd Director application. They are all extensions of the JDK base class Exception.

Exception Class	Description
EboException	This is the base exception class for the exteNd Director framework. Contains subclasses related to subsystem processing.
EboRuntimeException	A wrapper around java.lang.RuntimeException. Used for unexpected failures and instances when you cannot access the **throws** clause of a method's signature at runtime.
EboApplicationException	Indicates an error in the application. Typically, this signifies a programmer error such as an incorrect usage of an API method, illegal argument values, and so on.

Avoiding errors

The most common exception is the NullPointerException. To avoid it, be thorough about checking return values for null. For many exteNd Director methods, a return value of null is a valid response and tells you that the object you want does not exist.

In these situations, you should always make sure you have a valid object:

Situation	What to check
Getting values from the whiteboard or session	Did a value actually exist for the specified key?
Getting input data from forms	Did the parameter you tried to retrieve exist? If not, should you tell the user about the missing data or supply a default value? Was the value the user entered appropriate? If not, you need to tell the user what is correct.
Cached objects	Has the object you want been purged? If so, you need to reconstruct it.
Restricted objects	Does the current user have rights to the requested object? If not, you need to tell the user why the operation can't proceed.
User profile and preference data	Does a value exist for a profile key? If not, do you need to write one for the next access?

Catching errors

When you call methods in your application code that declare exceptions, you must enclose them in try/catch/finally blocks, as you would in any Java application. For runtime exceptions, you can use try/catch/finally blocks wherever the application's state would become invalid if an error occurs.

When your code catches an exception, you should display information about the problem to users so they know what is not working. For ideas, see “Displaying messages” on page 123. You should also do something appropriate to return the application to an error-free state. Don’t code empty catch blocks or simply log the exception.

Don’t do this These examples will not help the application or the user repair a problem:

```
try
{
   // code that could throw an exception
}
catch (Exception e)
{ }

try
{
   // code that could throw an exception
}
catch (Exception e)
{
   System.err.println("Exception: " + e.toString() );
   e.printStackTrace(context.getLocale());
}
```

A better way Do write catch blocks that prepare a message for the user and clean up whatever failed. The catch block can also re-throw the exception so that the calling method handles it.

These catch blocks catch two different exceptions and store an error message in the session. When the application finds a message in the session for the error key, it displays the message to the user. Constants identify the keys used for the session data.

```
catch (EboUnrecoverableSystemException e)
{
  // send trace to server console
  e.printStackTrace(context.getLocale());
  // save error message for later display
  String errMsg = e.getMessage(context.getLocale());
  m_portalSession.setValue(
       COMP_KEY, ERROR_MESSAGE_KEY, errMsg);
}
catch (Exception e)
{
  // send trace to server console
  e.printStackTrace(context.getLocale());

  // get the appropriate error message for later display
  ResourceBundle myResources =
   ResourceBundle.getBundle("MyResources", context.getLocale());
  String errMsg = myResources.getString("ERR_CAT_UNKWN");
  m_portalSession.setValue(
       COMP_KEY, ERROR_MESSAGE_KEY, errMsg);
}
```

 For information about the resource bundle shown here, see “Displaying errors in the user’s language” on page 123.

TIP: Instead of sending stack traces to the server console directly, you can use the exteNd Director logging facility, which identifies the output in useful ways. For information, see Chapter 15, “Logging Information”.

Displaying messages

One useful way to think about messages is to distinguish between messages that help users complete a task versus messages that tell them about error conditions out of their control.

Application code Your application logic might display these two types of messages in different ways. For example, it might display helpful messages right on the form that the user is filling out. To display error messages, you might replace the form as the content and explain why the form can't be displayed.

Pages A page can display errors by redirecting the browser to an error page that explains the error. JavaServer Pages provides a mechanism for displaying error pages. When an error occurs, the JSP page redirects the browser to an error page that you've created. For information on how to design error pages for JSP pages, see documentation from Sun Microsystems (java.sun.com).

Parent exceptions

If there has been a chain of exceptions that the application has re-thrown, the most recent exception may not have the most relevant message. You can get messages for all the exceptions that have occurred by checking the parent exceptions.

Example of examining parent exceptions In this example, the type of exception being caught could be any of the framework exception classes. The code concatenates all the messages for any exceptions that have been wrapped and re-thrown:

```
catch (EboException e)
{
  StringBuffer msg == "";
  java.lang.Throwable parent = e;
  while (parent != null)
  {
    msg.append( parent.getMessage(context.getLocale() + "\n");
    parent = EboException.getParentException(parent);
  }
}
```

Displaying errors in the user's language

To make your exteNd Director application useful internationally, you will want to display messages in the language of the user. You can get the user's locale from the EbiContext object with the getLocale() method. You can use it to get appropriately translated error messages:

- For exceptions, call getMessage() with a locale argument. exteNd Director provides localized versions of exception messages.

  ```
  e.getMessage(context.getLocale())
  ```

- Use resource bundles for storing message strings and access them using the user's locale. For each locale you want to support, you can provide a set of translated messages in subclassed bundles. For information, see java.util.ListResourceBundle and the example below.

You can get the user's locale from the EbiContext object with the getLocale() method.

Example of using resource bundles

The MyErrorsResource class has one string in its array of messages. A second class provides the same string in French. You can provide additional subclasses of ListResourceBundle in all the languages you want to support. The name of each uses the name for your default bundle plus a language code. To find out more about locales and language codes, see java.util.Locale.

```
class MyErrorsResource extends ListResourceBundle {
  public Object[][] getContents() {
      return contents;
  }
  static final Object[][] contents = {
  // LOCALIZE THIS
      {"ERR_CAT_UNKWN",
       "An unknown error has occurred while attempting to save the category
information."},  // unknown error for categories
  // END OF MATERIAL TO LOCALIZE
  };
 }
 //====================
 class MyErrorsResource_fr  extends ListResourceBundle {
  public Object[][] getContents() {
      return contents;
}
  static final Object[][] contents = {
  // LOCALIZE THIS
{"ERR_CAT_UNKWN",
       "Une erreur inconnue a arrivé en tentant à épargne l'information de
catégorie."},  // unknown error for categories
// END OF MATERIAL TO LOCALIZE
  };
 }
```

To access the string for a particular error situation, you need to know the key you have associated with the string. When you get the bundle, the JDK uses the locale to find the right version of the bundles you have provided. If no bundle matches the locale, the default bundle is used:

```
ResourceBundle myResources = ResourceBundle.getBundle(
   "MyErrorsResources",context.getLocale());
String msg = myResources.getString("ERR_CAT_UNKWN");
```

12 Working with Scoped Paths and XPaths

This chapter describes scoped paths and how to use them in your exteNd Director applications. It has these sections:

📖 For information about accessing scoped paths in the Pageflow and Workflow Modelers, see the chapter on the Pageflow Modeler in the *Pageflow and Form Guide* or the chapter on the WorkflowModeler in the *Workflow Guide*.

About scoped paths

Scoped paths allow you to quickly access data in your exteNd Director applications. The term *scope* refers to the state of the data, or data persistence. Data is *nonpersistent* if it is available for a single user session. Data is *persistent* if it is available to one or more users over multiple sessions and, potentially, to external applications.

An exteNd Director application has access to a number of different nonpersistent and persistent scopes. An HTTP Session is an example of a nonpersistent scope, and the exteNd Director resource set is an example of a persistent scope. The *path* part of a scoped path is simply the location of the data within a particular scope.

exteNd Director includes a group of predefined scopes that are available from the Pageflow and Workflow Modelers and through the Scoped Path API.

Advantages of scoped paths

Scoped paths make it easy to access and manipulate data:

- You don’t need to write data access code
- You can write link expressions on scoped data to redirect pageflows and workflows
- You can easily copy data from one scope to another

About XPaths

Many scoped paths support XPath-based navigation of XML documents (see “Predefined scopes” on page 126). XPath is a language defined by the World Wide Web Consortium (W3C) for addressing parts of an XML document.

You can append an XPath expression to a scoped path that points to an XML document. The Scoped Path dialogs in the Pageflow and Workflow Modelers provide a link to the Scoped Path Navigator tool for this purpose.

📖 For more information about the XPath Navigator, see “Using the Scoped Path and XPath Navigators” on page 139.

📖 For information about XPath, go to http://www.w3.org/TR/xpath20/

Predefined scopes

exteNd Director provides a set of predefined scopes with path syntax based on standard file access and XPath. The table below is a summary of the features for each predefined scoped path. The table columns mean the following:

- **Scope**: the name of the predefined scoped path. Click the name for more information.
- **Path**: the scoped path syntax, with available options.
- **XPath**: if marked, this scoped path supports XPath. For more information, see “Using the Scoped Path and XPath Navigators” on page 139.
- **Read**: if marked, this scoped path is readable. This means that the scoped data is available on a “copy from” function in the Pageflow and Workflow Modelers.
- **Write**: if marked, this scoped path is writable. This means that the scoped data is available on a “copy to” function in the Pageflow and Workflow Modelers.

  📖 For more information, see “Copying scoped paths” on page 136.

Scope	Path	XPath	Read	Write
Application scope	a path	●	●	●
Artifact scope	<dir>		●	
	cm://		●	
	war://		●	
CM scope	a path	●	●	●
Document scope	an xml document		●	

Scope	Path	XPath	Read	Write
Flow scope	document	•	•	•
	property		•	•
	exception		•	•
	documentproperty		•	•
	lock		•	
	unlock		•	
	persistent		•	
Format scope	Decimal		•	
	Date		•	
Log scope	LogID/Level			•
Portal scope	Url/Login		•	
	Url/Logout		•	
	Url/NewUser		•	
	Url/Resource/ a uri		•	
	Url/Context/ a uri		•	
	Url/Portlet/ a portlet name		•	
	an XML document	•	•	
PortletPreference scope	var name		•	•
Request scope	prop		•	
	param		•	
	attr		•	•
	api		•	
	cookie		•	
ResourceBundle scope	a string		•	
ResourceSet scope	directoryPath/elementKey	•	•	
Response scope	cookie			•
	render			•

Scope	Path	XPath	Read	Write
Session scope	a path	●	●	●
String scope	a string		●	
User scope	fname		●	
	lname		●	
	email		●	
	id		●	
	attr/key		●	●

Application scope

Description

The **Application scope** is an application-wide scope for the HTTP session. Objects stored in this scope are available to any other object that reside in the same portlet application and that handles a request in the same session. For the Pageflow Modeler, this includes any other flow activities (including JSP and Servlet activities) and any other portlet contained in the same session.

Path

`Application/aKey/path/xpath`

Examples

Write (Copy to) example:

```
Session/doc/mydoc --> Application/docout/doc1
```

Read (Copy from) example:

```
Application/prop/aString --> Flow/property/prop1
```

Artifact scope

Description

The **Artifact scope** is an application-wide scope for persistent data stored in any of the following:

- The exteNd Director Content Management subsystem
- The file system of the host machine
- The application WAR

This scope supports XPath navigation to XML documents. Artifact scopes are available on the “copy from” side of a scope copy in the Pageflow and Workflow Modelers.

TIP: It is more efficient to use the ResourceSet scope and CM scope scopes rather then the Artifact scope whenever possible.

Path

```
Artifact/cm://path/xpath
Artifact/war://path/xpath
Artifact/dir/path/xpath
```

Examples

Write (Copy to) example:

```
Flow/document/mydoc --> Artifact/c:/applications/docs/mydoc1.xml
```

Read (Copy from) example:

```
Artifact/c:/applications/docs/mydoc1.xml --> Flow/document/mydoc
```

CM scope

Description The **CM scope** is an application-wide scope for persistent data stored in the Content Management Subsystem. You must enter a known repository and path. On a “copy to” function (writable) the document is written to the specified location in the PUBLISHED status. This scope supports XPath expressions.

Path `CM/myrepository/mypathtodoc/xpath`

Examples Write (Copy to) example:

```
Flow/mydoc/doc --> CM/companydoc/reports/report1.xml
```

Read (Copy from) example:

```
CM/companydoc/reports/report1.xml --> Flow/mydoc/doc
```

Document scope

Description The **Document scope** allows you to create a temporary XML document. In a Pageflow or Workflow application the Document scope is available for a single object in a flow application, and is valid only on the “From” side of a copy.

NOTE: The document must be a valid (well-formed) XML document.

Path `Document/<elem></elem>`

Example This example shows a series of scoped path copy operations that copy elements from the Document scope to the Session scope:

```
/Document/<books/> -> /Session/bookref
/Document/<book>Jungle book</book> ->/Session/bookref/books
/Document/<book>Catcher in the Rye</book> -> /Session/bookref/books
/Document/<book>Moby Dick</book> -> /Session/bookref/books
```

In the example shown above, you see how to create a document on the Session scope that contains multiple nodes with the same name. In the following example, you see an alternative way to do this:

```
/Document/<books/> -> /Session/bookref
/Document/<book/> -> /Session/Temp
/Request/param/title1 -> /Session/Temp/book
/Session/Temp -> /Session/bookref/Books
/Document/<book/> -> /Session/Temp
/Request/param/title2 -> /Session/Temp/book
/Session/Temp -> /Session/bookref/Books
```

Flow scope

Description.

The **Flow scope** is used within a pageflow or a workflow. In a pageflow the flow scope is available to all flow objects (including contained pageflows) for the portlet application session. In a workflow the flow scope is available to all flow objects (including contained pageflows) for the duration of the workflow process.

The Flow scope provides exclusive access to these objects:

Flow scope option	Description
document	XML document (this scope supports XPath expressions). 📖 See "Using the Scoped Path and XPath Navigators" on page 139.
property	A string variable
exception	A message type or other data to associate with an Exception activity 📖 For more information, see Exception Activity in the *Pageflow and Form Guide*.
document property	A property (metadata) associated with a document
The next three properties apply to a pageflow running within a workflow. 📖 For more information, see the section on pageflows in a workflow in the *Workflow Guide*.	
lock	Attempts to lock a workitem for the current user: ◆ 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: ◆ 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 **true** if the workitem is in a persistent state (workflow context) and **false** if not persistent (pageflow context).

Path

```
Flow/document/varName/xpath
Flow/property/varName
Flow/exception/type
documentproperty/varName
```

Examples

Write (Copy to) example:

```
Session/doc/mydoc --> Flow/document/doc1
```

Read (Copy from) example:

```
Flow/doc/doc1 --> CM/companydoc/reports/report1.xml
```

Format scope

Description. The **Format scope** creates a unique identifier at runtime that can be appended to another scope or referenced by other activites in a flow.

This scope supports two types of formatters. Both of these formatters use the patterns as defined in the associated java.text.formatter class:

Format scope option	What it does
Decimal	Generates a number in the specified format. Starts the get sequence at 0, and increments by 1 each time the getValue is called on the scope. This formatter does not allow for either initial value setting or a different increment setting. 📖 For available formats, see DecimalFormat in the *API Reference*.
Date	Generates a timestamp in the specified format. 📖 For available formats, see SimpleDateFormat in the *API Reference*. 📖 To generate localized timestamps, use one of the patterns defined in DateFormat (SHORT, MEDIUM, LONG, and FULL).

Path

```
Format/Date|Decimal/pattern
```

Examples

```
Format/Decimal/000.0#
Format/Date/yyyy.MM.dd
Format/Date/yyMMddHHmmsss
Format/Date/FULL
```

Copy example:

```
CM/myRepository/folder1/${/Format/Date/yyyy.MM.dd}.doc
```

Log scope

Description The **Log scope** allows you to access the exteNd Director logging facility—runtime logging facility that writes information in one of several standard logs or in your own custom logs. You decide what level of detail you want in the logs. The output of the logs displays on the server console. If you don't specify the logging level, the default value is used.

The Log scope is available on the "To" side of copy in the Pageflow or Workflow Modelers.

📖 For more information, see Chapter 15, "Logging Information".

Path

```
Log/LogIdentifier/Level
```

Examples

```
Log/EboFwLog
Log/EboCmLog/5
```

Copy example:

```
String/mystring --> Log/myLogIdentifier
```

Portal scope

Description

he **Portal scope** returns a fully qualified URL to a portal resource. This scope is valid only on the "From" side of a copy.

Portal scope option	Path option	Description
Url	Login	Returns PortalLogin portlet URL.
	Logout	Returns Logout URL (terminates the current user session).
	NewUser	Returns NewUser portlet URL
	Home	Returns user-defined default page. If not defined or if the user is **anonymous**, returns the portal default page URL.
	Resource	Returns the ResourceSet URI.
	Context	Returns the Director application context URI.
	Portlet	Returns a fully qualified URL to a portlet.
Keyword See Using the Keyword option below	Navigation/ContainerPages	Displays navigation for container pages
	Navigation/SharedPages	Displays navigation for shared pages
	Navigation/PersonalPages	Displays navigation for personal pages
	Navigation/QuickLinks	Displays navigation for quick links

Path

```
Portal/Url/Login
Portal/Url/Logout
Portal/Url/NewUser
Portal/Url/Home
Portal/Url/Resource/ a uri
Portal/Url/Context/a uri
Portal/Url/Portlet/a portlet name
```

Examples

```
Portal/Url/Resource/images/DirectorWelcomeHeader.jpg
Portal/url/Context/PAC
Portal/Portlet/StockQuote
```

Using the Keyword option

Use the **Keyword** option to display the installed Navigation portlet by using the scoped path syntax in a pageflow activity. For example:

```
${Portal/Keyword/Navigation/SharedPages}
```

📖 For more information, see "Scoped path syntax in pageflow activites" on page 138.

Keyword accesses the XSL pattern used by the installed Header portlet and the Navigation portlet. These two portlets use the same portlet class to display the content of a preference named **layout**, in conjunction with a specific XSL style sheet. The actual content is determined by the context user.

📖 For more information see in the chapter about portal pages in the *Portal Guide*

PortletPreference scope

Description

The **PortletPreference** points to the current portlet preference value with the specified variable name. This scope is private to the portlet preferences settings for the current portlet session. Values in this scope persist for the user and the portlet instance.

 For more information, see the section on portlet preferences in the*Portal Guide*.

Path

```
PortletPreference/varName
```

Examples

Write (Copy to) example:

```
String/'300" --> PortletPreference/height
```

Read (Copy from) example:

```
PortletPreference/height --> Session/prop/height
```

See also

"Scoped path syntax in portlet preferences" on page 138

Request scope

Description

The **Request scope** represents the request object associated with a portlet or HTTP request. Request scope values are available on portlet action and render requests, as described in the table below.

 For information about how render and action requests are handled in pageflows, see the chapter on Working with Pageflows in the *Pageflow and Form Guide*.

The Request scope provides access to these values:

Request scope option	Description	Calls this method:
param	Returns the String object associated with the specified parameter. Available on render and action requests.	getParameter() on EbiRequest.
attr	Sets or returns the String object associated with the specified attribute. Available on render and action requests.	get/setAttribute() on EbiRequest
cookie	Returns the cookie value associated with the specified cookie name. Available on render and action requests. **NOTE:** This option returns an Object. If you are using it in a Pageflow it will not be available for a link expression or for rendering without further processing (in a Java activity, for example).	getCookieValue() on EboCookieUtil
api	Returns the value associated with the specified request method. Available on render requests only. In the Pageflow Modeler, select the method from the tree view.  See "Request API option and PersistMgr realm" below.	See methods on javax.portlet.RenderRequest

Request scope option	Description	Calls this method:
prop	Returns the HTTP request header for the specified property. Available on render and action requests. In the Pageflow Modeler, select the property from the tree view.	getHeader() on EbiRequest

Request API option and PersistMgr realm If you are using the PersistMgr realm configuration, the get() methods on the **api** option related to authenticated users—getAuthType(), getRemoteUser(), and getUserPrincipal()—will return null. This is because the request object does not get populated with authentication data by the application server when authentication is done through the PersistMgr.

You can get authentication information by instantiating a context object (com.sssw.fw.api.EbiContext) in your code. Use one of the createEbiContext() methods on EboFactory.

Request API option and iChain single sign-on To provide sign-on support for iChain, you may want to pull the user ID and password out of the request header. The getUserID() and getPassword() methods on the Request **api** option give you a way to do this.

Path

```
Request/param/varName
Request/attr/varName
Request/cookie/varName
Request/api/method
Request/prop/varName
```

Examples

Write (Copy to) example:

```
Session/mykey/locale --> Request/attr/locale
```

Read (Copy from) example:

```
Request/api/getLocale --> Session/mykey/locale
```

ResourceBundle scope

Description The **ResourceBundle scope** allows you scope to a defined Java resourceBundle.

Path

```
ResourceBundle/family/aKey
```

Examples

Copy example:

```
ResourceBundle/myResourcesFile/Key1 --> Session/localize/locale
```

ResourceSet scope

Description The **ResourceSet scope** is an application-wide scope for persistent data stored in your exteNd Director project ResourceSet.

NOTE: This scope is similar to the Artifact WAR scope but restricts access to known locations in the WAR's ResourceSet.

This scope supports XPath navigation to XML documents.

Path

```
/ResourceSet/path/xPath]
```

Example

```
ResourceSet/pages/category/pagetype --> Flow/property/ptype
```

Response scope

Description

The **Response scope** represents the response object associated with a portlet response.

Response scope values are available on both portlet action and render responses, unless specified otherwise in the table below. For information about how render and action responses are handled in pageflows, see the chapter on working with Pageflows in the *Pageflow and Form Guide*.

The Response scope provides access to these values:

Response scope option	Description	Calls this method:
cookie	Creates a cookie with the specified value and sets it in the browser. Available on render and action responses.	addCookieToResponse() on EboCookieUtil
render	Sets the specified String parameter for the Portlet render request. Available on render responses only.	setRenderParam() on ActionResponse

Path

`Response/cookie/varName`
`Response/render/varName`

Example

Copy example:

```
Session/param/myString --> Response/cookie/anAttribute
```

Session scope

Description

The **Session scope** is private to the portlet and its included resources for the current portlet session. Objects in this scope are namespaced to be unavailable to other Web components in the portlet application.

IMPORTANT: If you want to access session variables from a Pageflow JSP page or Servlet activity, you need to copy the data to the Application scope. For more information, see Application scope.

Path

`Session/aKey/path/xpath`

Examples

Write (Copy to) example:

```
CM/myrepository/mypathtodoc/pubdoc.xml --> Session/doc/mydoc
```

String scope

Description

The **String scope** allows you to create a temporary string. In a Pageflow or Workflow application the String scope is available for a single object in a flow application, and is valid only on the “From” side of a copy.

Path

`String/aString`

Example

Copy example:

```
/String/I am here -> /Session/myparam
```

User scope

Description

The **User scope** provides access to the logged in user and user attributes. The User scope provide access to these values:

User scope option	Value
fname	The first name attribute of the logged in user
lname	The last name attribute
email	The email attribute
id	The userID attribute
attr	The value of a specified custom attribute.

Path

```
User/fname
User/lname
User/email
User/id
User/myattr
```

Examples

Write (Copy to) example:

```
Session/attrubute --> User/attr/attribute1
```

Read (Copy from) example:

```
User/email --> Session/contact
```

Copying scoped paths

The Pageflow and Workflow Modelers provide different ways to copy data from one scope to another at selected points in the flow application. For example, you can use scoped paths to write logical expressions on links and to copy data to and from different scopes.

NOTE: The copy function in the Pageflow and Workflow Modelers copies one value/object per copy. If you want to work with collections of objects, like scoping to a node in a DOM, you need to handle the parsing in a Java Activity or XSL.

Copy options

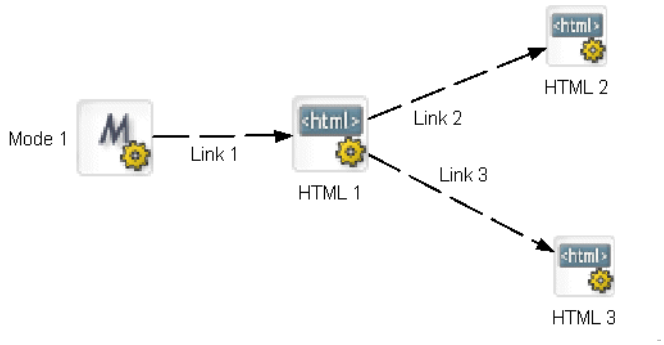
You can copy a scoped path before or after the execution of a Pageflow or Workflow activity, or after a link is followed. The copy option you use depends on the logic of your flow. Here is how scoped paths are handled for each copy option:

Copy option	Flow processes this option	Use this option when
Before this activity	After source link is evaluated and before this activity is executed	This activity has one source link or all source links will share the scoped data
After this activity	After this activity is executed and before any destination link is evaluated	The next activity has one source link or all of its source links will share the scoped data
Link	After this link is followed and before any destination activity is executed	The destination activity has more than one source link and each link uses different scoped data

When to copy on activites

You can copy scoped paths on activites whenever there is no conflict between two or more scoped paths going to the target activity. This is appropriate when:

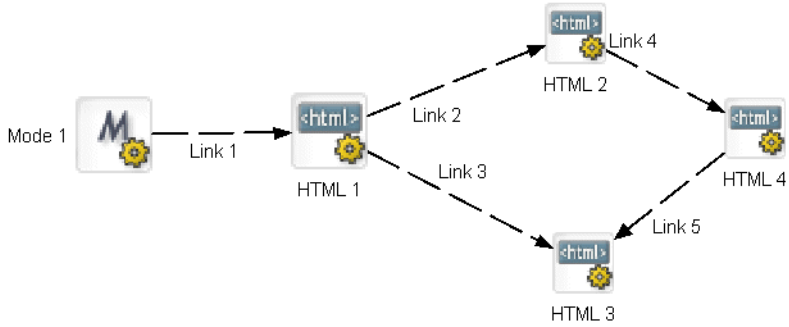
- A single link is associated with each activity (see example below)
- Multiple links are associated with an activity, and the scoped data on each source activity is identical.



Suppose you want to return different pages results based on Link 2 and Link 3. In this scenario, since there is no potential conflict between two links going to a single activity, you could either copy the data on Link 2 and Link 3 or copy the data before the execution of HTML 2 and HTML 3.

When to copy on links

You need to copy data on links when you have more that one link going to a single activity and you have different scopes associated with each link. Consider this scenario:



Here you have two links targeting HTML 3. If you want different data associated with this activity—that is, if you want the data to be dependant on the link source—you would need to copy the data on the links.

NOTE: Although you could have two “copy before” scopes on HTML 3, the flow engine has no way of distinguishing them by source link.

Using the scoped path substitution syntax

Scoped paths include a substitution syntax that can be included in certain application elements and resolved at runtime.

Scoped path syntax in pageflow activites

Scoped path support includes substitution syntax for accessing a path directly from HTML and XForm activities in pageflow applications.

NOTE: The standard syntax applies **only** to objects running in exteNd Director Pageflows.

The syntax is:

```
scopedpath?my_scoped_path/scopedpath
```

For example, in an HTML page:

```
<P>scopedpath?Request/attr/myVarName/scopedpath </P>
```

Substitutes the string value of *myVarName* for the scoped path syntax.

Dynamic resolution in scoped paths

You can do dynamic resolution of scoped paths by appending the following syntax to an existing scoped path:

```
${myPath}
```

For example, to generate a dynamic document in the Content Management subsystem with the current timestamp as the file name, you could append the Format scope to the CM scope like this:

```
CM/myRepository/folder1/${/Format/Date/yyyy.MM.dd}.doc
```

Scoped path syntax in rules

You can use the dynamic substitution syntax ${path} in the Rules subsystem to access scoped paths. This function is available in any condition or action that has *template fields*.

For more information, see Installed Conditions and Installed Actions in the *Rules Guide*.

You can also nest a dynamic resolution within a rule using this syntax:

```
{$CM/folder/${Format/Date/<pattern>}}.doc
```

Scoped path syntax in portlet preferences

The dynamic substitution syntax can also be used to represent a value in a portlet preference descriptor. This example substitutes the value of the String scoped path for a portlet preference value:

```
...
<text-color> {$String/red} </text-color>
...
```

About the Scoped Path API

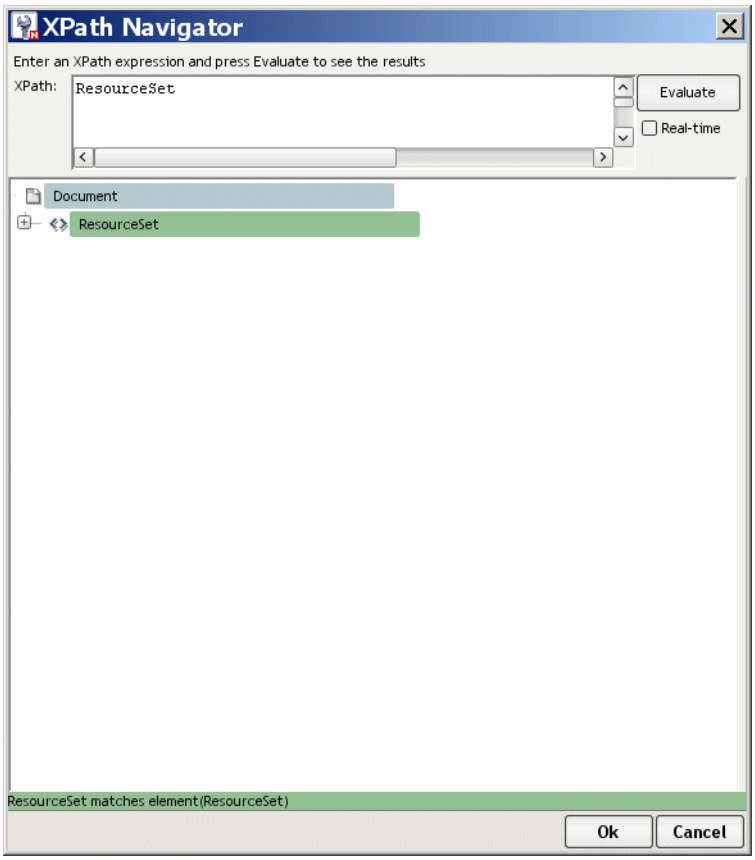
You can use the Scoped Path API to access scoped paths directly from your application code—for example, from a Java, JSP, or Servlet activity. Here are the principal classes for accessing scoped paths:

Package and class	Provides
com.sssw.fw.api.EbiScopedPath	Methods for accessing an instance of a scoped path
com.sssw.fw.factory.EboFactory	createScopedPath method for getting a scoped path instance

Using the Scoped Path and XPath Navigators

NOTE: The Scoped Path and XPath Navigators are essentially the same tools with slightly different features. This section covers the Scoped Path Navigator. For practical purposes, it applies to the XPath Navigator as well.

The Scoped Path Navigator is available from the Pageflow and Workflow Modelers and allows you to select a scoped path using a tree view. When you select one of the predefined scoped paths, the Scoped Path Navigator appears in a separate window:



The tree view also provides facilities for building XPath expressions.

➢ **To select a path in the Scoped Path Navigator:**

1 Use the tree view to select the path.

The path you select is reflected in the text box at the top. You can type the path directly in the text box instead of using the tree view.

2 Click **OK** at the bottom of the Navigator window.

This returns you to the Access dialog.

Creating XPath expressions

Some of the scoped paths support XPath expressions (see "Predefined scopes" on page 126). The Scoped Path Navigator supports some of the XPath expressions as defined by W3C. For usage details, see http://www.w3.org/TR/xpath20/.

IMPORTANT: The Scoped Path Navigator is capable of returning only one document element for each scoped path expression. If you use an XPath function that specifies more than one element, only the first element in the group will be returned.

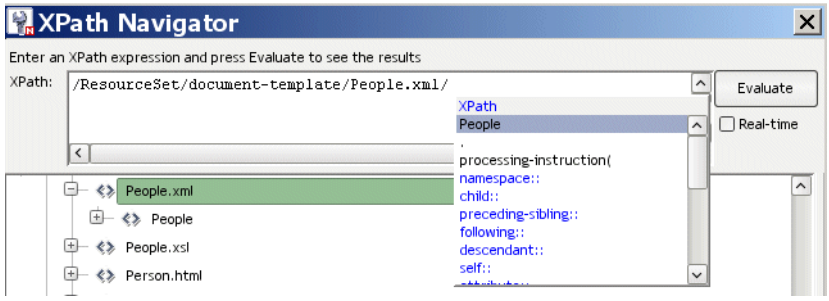
➢ **To create an XPath expression:**

1 Navigate to an element (typically an XML document) where you want to enter an XPath expression.

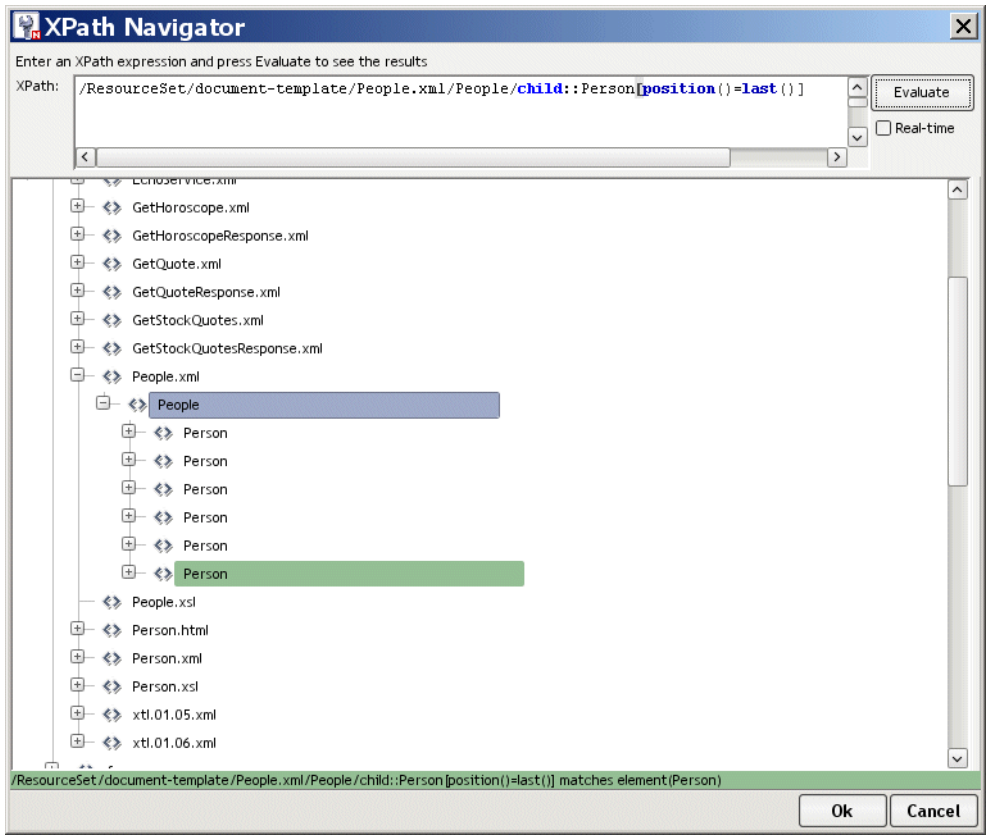
2 Select **Real-time** to have your expression evaluated automatically as you enter it, or select **Evaluate** to evaluate manually. The result of the evaluation displays at the bottom of the window.

NOTE: You can enter the expression in the editor field without using the XPath facilities described below.

3 Navigate to an XML document using the tree view or the editor and enter a valid XPath delimiter: a slash / character, an open bracket [, or an open parenthesis (. The Navigator displays a dropdown list of elements and functions available at the point you enter the delimiter. For example:



4 To add an item, select it from the dropdown, and type in an appropriate value when applicable. For example:



5 Repeat as needed until the expression is complete and valid.

6 Click **OK** at the bottom of the Navigator window.

This returns you to the access dialog.

TIP: You can right-click an element to access other XML editing facilities. For more information, see the chapter on XML Editors in *Utility Tools*.

13 Working with Events

This chapter describes the exteNd Director event model and event-handling concepts. It has these sections:

About the exteNd Director event model

The exteNd Director event model is an extension of the event listener/producer model in Java. An *event* is a lightweight notification object that contains information relevant to one or more *event listeners*. The listener responds to the event in an appropriate way. The primary user of events in exteNd Director is the Content Management (CM) subsystem, which defines event objects for various CM operations. For example, you can register a listener for the "document added" operation that generates an event carrying information about the document's author, title, and other data. You can handle this event in any way you choose, like making a log entry, writing to a print stream, or e-mailing interested parties. exteNd Director provides an extensible event framework with a full set of predefined events for CM, WebDav, and CM task management operations.

This chapter describes basic event concepts in exteNd Director. For information about event support for specific subsystems, see the following:

Subsystem or function	Information about events
CM subsystem	Working with content management events
CM tasks	Working with CM Task events
WebDAV	Working with WebDAV events

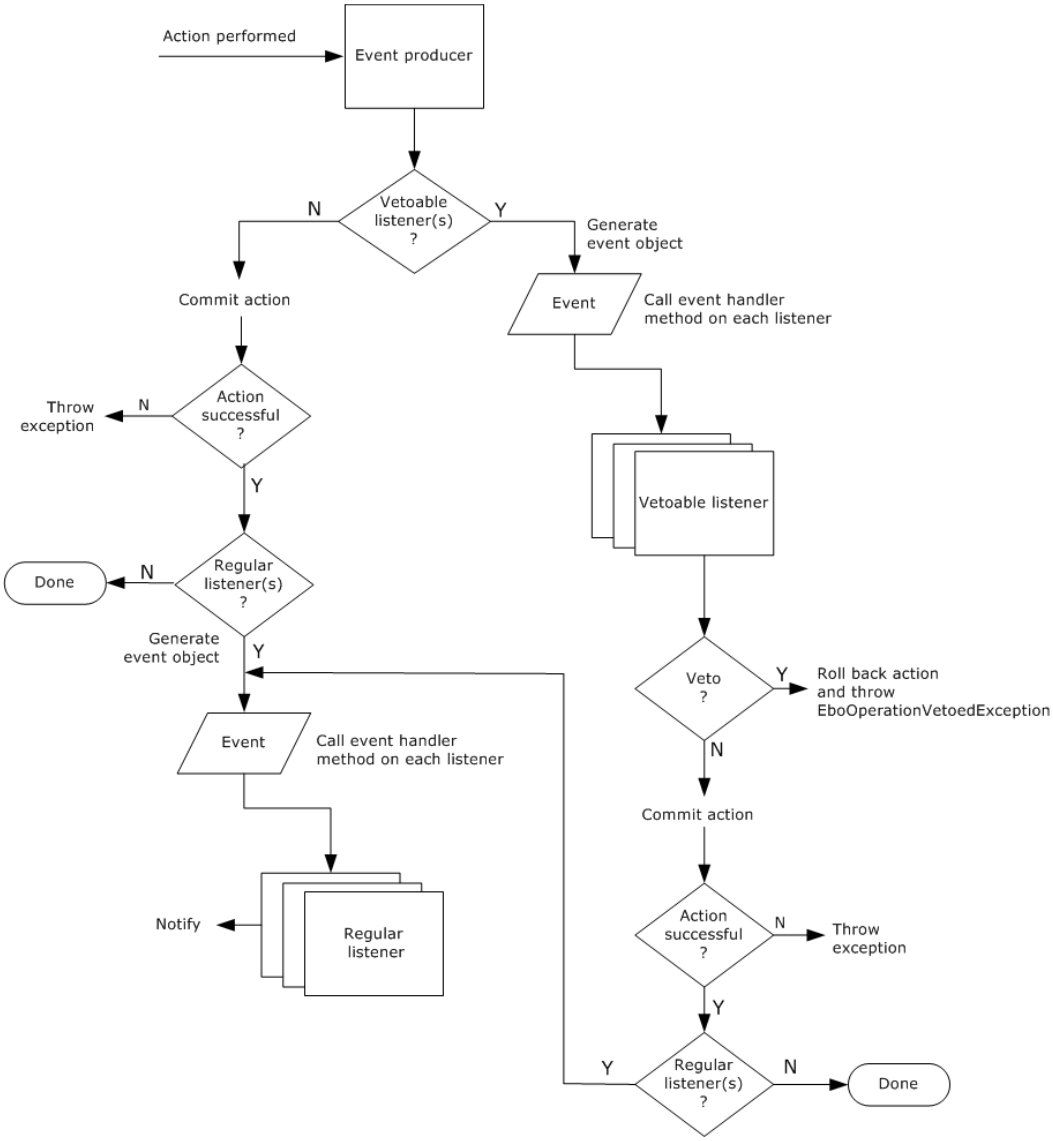
Event model object types

The exteNd Director event model consists of these types of objects:

Event object type	What it does
State change event	Represents any change that may occur within the scope of an application. Changes are generally related to object life cycles and operations. The state change event contains: ◆ A unique integer event ID ◆ A state change ID ◆ a description of what the event represents ◆ Access to the context (EbiContext)
State change producer	Monitors specific actions and generates corresponding state change events.
State change listener	Registers with one or more state change producers and listens to all or a specified subset of the events as they are generated.
Vetoable event listener	Registers with one or more state change producers and has the ability to veto (nullify) an action that an event represents. Vetoable event listeners are always notified by the state change producer **before** any state change listeners, as well as before the action is committed.

Event handling

Event handling is the processing of a specific event by a listener after the listener's *event handler method* is called by the event producer. In this context, the event listener functions as the *event handler object*. Here is a flow diagram of the event handling process. The flow assumes an event associated with a database transaction, like a CM repository update:



Event handling with regular listeners

In this scenario, several regular listeners have been registered with an event producer. An action in the event domain of the producer has occurred—such as a user adding a document to the CM repository. Here is the event-handling sequence:

1 After verifying that there are no vetoable listeners registered for this event, the event producer performs the action requested.

2 If the action fails for some reason, the producer handles the exception.

3 If the action succeeds, the event producer instantiates an event object and populates it with relevant information.

4 The event producer calls the stateChanged() event handler for each listener, passing in the event object.

NOTE: The event producer notifies the listeners in the order in which they were registered.

5 As the method is called, each listener performs any specified notification, such as e-mailing an interested party.

Event handling with vetoable listeners (with veto)

In this scenario, several vetoable and regular listeners are registered for the event, and the action is vetoed. Here is the event-handling sequence:

1 The event producer instantiates an event object and populates it with relevant information.

2 The event producer calls the vetoableStateChanged() event handler for each vetoable listener.

3 Since the event is vetoed, no action is performed by the producer and no regular listeners are notified.

Event handling with vetoable listeners (with no veto)

In the final scenario, several vetoable and regular listeners are registered for the event, and the action is **not** vetoed. Here is the event-handling sequence:

1 The event producer instantiates the event object.

2 The event producer calls the vetoable stateChanged() event on each vetoable listener.

3 Since no listener vetoes the action, the event producer performs it.

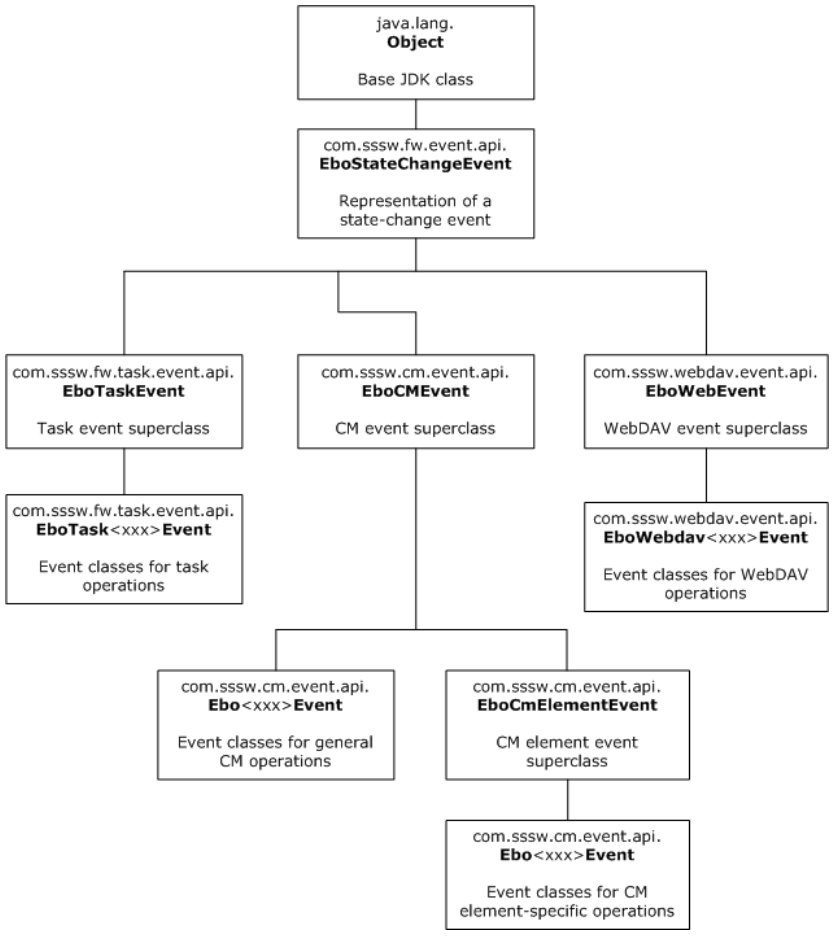
4 Assuming the action does not fail, the producer iterates over the regular listeners by calling their stateChanged() methods.

About the Event API

This section is an overview of the exteNd Director Event API.

Event classes

This diagram shows the class hierarchy for the event objects:



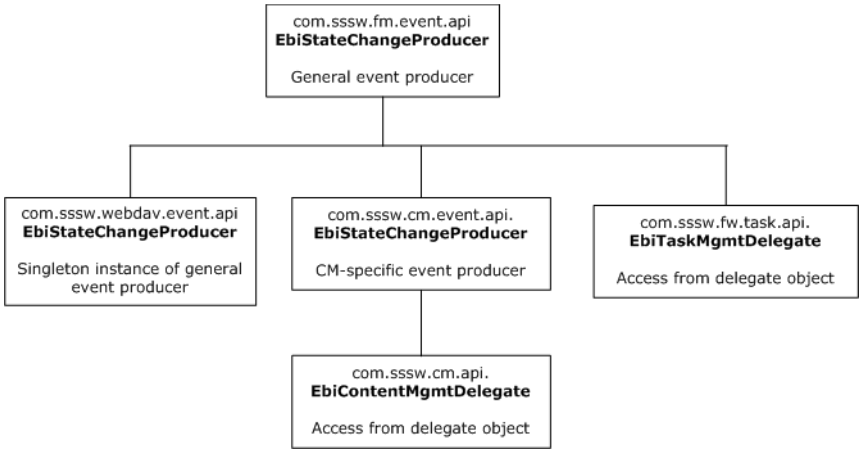
Event constants

Also (not shown in the diagram), the Event API includes class constants related to event monitoring:

- com.sssw.cm.event.api.EbiConstants
- com.sssw.fw.task.event.api.EbiConstants
- com.sssw.webdav.event.api.EbiConstants

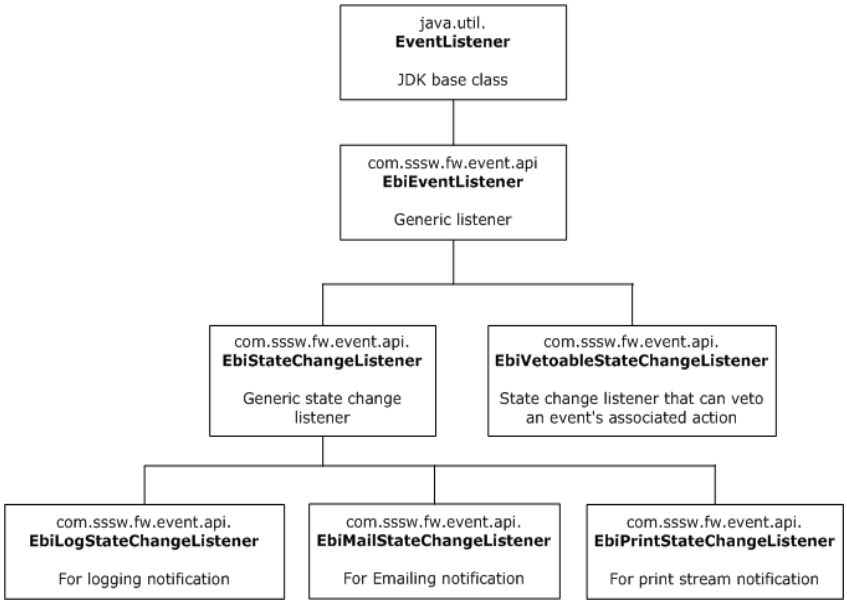
Producer interfaces

This diagram shows the class hierarchy for the event producer interfaces:



Listener interfaces

This diagram shows the class hierarchy for the event listener interfaces:



Creating and registering listeners

To implement events in your application, you first need to determine what events you are interested in and how you want to handle them:

- **Decide what types of operations you want to monitor**. In the case of content management elements, decide what element (directories, documents, document types etc.) in the repository you are interested in.
- **Figure out the logic you need to handle the event**. Typically you'll want to notify an interested party, record the event in a log, or respond in some other manner.

- **Determine whether there are conditions under which you want to veto an operation represented by an event**. In these cases use a vetoable listener.

Implementing your event scheme involves these steps:

1 Instantiate a listener to handle the event.

TIP: For reusability you can create a separate listener class that implements one or more of the event listeners.

2 Register the listener and events using one of the add listener methods on the event producer.

Using notification listeners

Event handling usually involves some kind of notification, like making a log entry or e-mailing an interested party. exteNd Director provides the following support for event notification:

Notification listener	What it does
EbiMailStateChangeListener	Sends an e-mail to specified parties
EbiLogStateChangeListener	Writes to a specified log
EbiPrintStateChangeListener	Writes to a specified print stream.

You can get a notification listener using the appropriate factory method. For example:

```
EbiMailStateChangeListener listener =
com.sssw.fw.event.factory.EboFactort.getMailStateChangeListener();
```

Creating a notification listener

This example shows how to create a class that implements an EbiMailStateChangeListener by delegating to the default listener:

```
public class MyClass implements com.sssw.fw.event.api.EbiStateChangeListener {
     protected EbiMailStateChangeListener m_scl;

     public void stateChanged(EboStateChangeEvent event) {
     try
     {
          EbiMailStateChangeListener scl = getScl();
          scl.setMsgText(…);
          // provide other settings ...
         scl.stateChanged(event);
    }
catch (Exception ex) {
     // handle exceptions here
   }
}
protected EbiMailStateChangeListener getScl() throws EboFactoryException {
     if (m_scl == null)
          m_scl =
com.sssw.fw.event.factory.EboFactory.getMailStateChangeListener();
     return m_scl;
   }
}
```

Using a vetoable listener

To veto an operation, you define the veto condition in the vetoable listener's vetoableStateChanged() event and return **false**. The event producer responds by vetoing the operation and throwing com.sssw.fw.exception.EboOperationVetoedException.

Because EboOperationVetoedException is a runtime exception, it is not included in the throws clause of the operation methods. However, you must handle the exception somewhere in your code. 1For example: if you added a vetoable listener that included the content management **add document** event, you would need to handle the exception for the addDocument() method, as shown here:

```
try
{
     EbiContentMgmtDelegate cmgr = ...
     EbiAddDocumentParams params = ...
     EbiDocument doc = cmgr.addDocument(context, params);
     .....
}
catch (EboUnrecoverableSystemException ue)
{
    // handle unrecoverable system exception
}
catch (EboSecurityException se)
{
   // handle security exception
}
catch (EboItemExistenceException iee)
{
   // handle item existence exception
}
catch (EboOperationVetoedException ove)
{
   // handle operation vetoed exception
}
catch (Exception e)
{
   // handle any other exception
}
```

Creating a custom state change listener

To create a listener you need to implement one or more of the listener interface(s) in your application code.To create a regular (non-vetoable) listener, provide an implementation for the stateChanged() method, as shown here:

```
public class MyClass implements
  com.sssw.fw.event.api.EbiStateChangeListener {

public void stateChanged(EboStateChangeEvent event) {
     // perform action, inspect event, and notify...
   }
}
```

IMPORTANT: You must provide implementations for the four methods on the super class EbiEventListener (extended by EbiStateChangeListener). This applies to creating a vetoable listener as well.

Creating a vetoable listener To create a vetoable listener, provide an implementation for the vetoableStateChanged() method, as shown here:

```
public MyClass implements EbiVetoableStateChangeListener
{

     public boolean vetoableStateChanged(EboStateChangeEvent event)
    {

      // Inspect event:
      // If vetoed, return false
      // If not vetoed return true and perform action
    }
}
```

IMPORTANT: If an action or operation is vetoed, the event producer throws a runtime exception called EboOperationVetoedException. See "Using a vetoable listener" on page 149.

Registering for events

You register for events using one of the add listener methods on EbiStateChangeProducer or one of its subclasses.

When you add an event listener you can specify a range of events for which you want to register in a Java BitSet, using the addStateChangeListener() or addVetoableStateChangeListener() method. For example:

```
EbiStateChangeProducer producer = new EbiStateChangeProducer()
BitSet events = new BitSet();
events.set(MyEvent.getEventID());//interested in MyEvent
events.set(MyEvent2.getEventID());//interested in MyEvent2
producer.addStateChangeListener(events, MyListenerClass);
```

The event IDs for content management, WebDAV, and task management are defined as constants in the respective subsystem API packages. The com.sssw.cm.api also provides some helper methods for populating the BitSet.

📖 For more information, see the section on using the event helper class in the *Content Management Guide*.

Creating custom events and producers

You can extend the Event API to write your own state change event and event producers.

Creating a custom event producer

You can write your own event producer that uses custom versions of the addStateChangeListener() method. To create a state change producer, you must implement the com.sssw.fw.event.api.EbiStateChangeProducer interface.

The example that follows shows how to delegate to the default event producer. It uses a factory method to get the default state change producer, adds a description and log, and provides an implementation of addStateChangeListener():

```
public class MySCP implements
  com.sssw.fw.event.api.EbiStateChangeProducer {
     protected EbiStateChangeProducer m_scp;
     public MySCP() {}
     public boolean addStateChangeListener(BitSet events, EbiStateChangeListener listener) {
          getScp().addStateChangeListener(events, listener);
     }
     // ….
     // other EbiStateChangeProducer methods, implemented with addStateChangeListener()…
     // ….
     protected EbiStateChangeProducer getScp() {
          if (m_scp == null) {
               m_scp = com.sssw.fw.event.factory.EboFactory.getStateChangeProducer();
               m_scp.setScpDescription("My state change event producer");
               m_scp.setScpLog(EboLogFactory.getLog(MYLOG));
}
return m_scp;
     }
}
```

Creating a custom event

A custom state change event must extend com.sssw.fw.event.api.EboStateChangeEvent. You need to implement the following two methods that are marked abstract in the superclass:

```
public abstract int getEventID();
public abstract String getVerboseDescr();
```

NOTE: The exteNd Director Framework API reserves the value range 1 through 8000, so you must use another value for your event ID.

14 Working with Data Caches

This chapter describes how to handle data caching in exteNd Director applications. It has these sections:

- About data caching
- Request object caching
- Session-level caching
- server-lifetime caching

About data caching

Data caching provides a way to manage the temporary storage of application data. The most common purposes for managing data caches are:

- Improving performance by eliminating system resources needed to retrieve commonly used data
- Recovering data in the event of session or server failure

exteNd Director supports data caching at different application levels, including HTTP request, application session, portlet application, and server-lifetime (for server clusters).

About the Cache Manager

The EbiCacheManager interface allows you to manage session-level and server life-time data. It provides two cache mechanisms:

- Object cache container
- Memory and disk cache containers

Object cache container

The object cache container stores all objects in memory, regardless of the size of the content. It is used when the putObjectInCache() method is called from the cache holder object, described in the next section (About the cache holder). The object cache does not require cached objects to be serializable.

Use the object cache container if you are not primarily concerned about the size of the cached content and/or if you need the ability to store nonserializable objects.

How it works The object cache container uses the maximum number of objects specified in the cache configuration to determine when to remove the least-used objects from its cache. In cases where the maximum number of objects has not been reached but memory is running low, the object cache container removes the least-used objects from its cache.

Memory and disk cache containers

The memory cache and disk cache container mechanism stores objects either in memory or on disk depending on the size of the object. This caching mechanism requires cached objects to be serializable. It is used when the putValueInCache() method is called from the cache holder object. (See “About the cache holder” on page 154.)

Use this implementation when you want to remove larger objects from memory and cache them on disk, keeping in mind that the objects must be serializable.

How it works The memory and disk cache containers use a maximum byte-size value to determine whether to cache items in memory or larger items on disk. You can use the Cache Manager to configure this value.

NOTE: The object cache and the memory cache each maintain their own cache in memory, and are managed separately.

Configuring the Cache Manager

You can reconfigure the default Cache Manager settings using one of these methods:

- Cache settings in the DAC
- ContentCache settings in config.xml

Cache settings in the DAC You can use the Cache settings in the Director Administrator Console (DAC) to configure the Cache Manager. Values set in the DAC are not stored beyond the current server session.

For more information, see Chapter 22, “Using the General Configuration Section of the DAC”.

ContentCache settings in config.xml To make persistent changes, edit the ContentCache settings in config.xml, located in the FrameworkService-conf directory in your project. For a description of the settings, see Chapter 22, “Using the General Configuration Section of the DAC”.

About the cache holder

The EbiCacheHolder interface defines caching methods for the session-lifetime cache holder (EbiSession) and the server-lifetime cache holder (EbiSrvLifetimeCacheHolder). A cache holder can store content in the object cache container or the jointly managed memory and disk cache containers, as described in the preceding section. (About the Cache Manager.)

To access the session cache, call the appropriate method in EbiSession. To access the server-lifetime cache, call the appropriate method in EbiSrvLifetimeCacheHolder. Here are some of the methods (inherited form EbiCacheHolder) on both of these objects:

Cache holder methods	Usage
putObjectInCache() getObjectInCache() removeObjectInCache()	Put, get, and remove objects from the object cache. Cached objects do not need to be serialized.
putValueInCache() getValueInCache() removeValueInCache()	Put, get, and remove objects from the memory and disk cache containers. Cached objects must be serialized.

Request object caching

You can access request parameters directly from the application's request object or use temporary values in the context object.

NOTE: If you need to cache request values for a session, for session-level fail over for example, you must store them on the whiteboard. For more information, see "Using the whiteboard" on page 156.

Request object attributes

Request attributes are stored in the associated servlet or portlet request object: HttpServletRequest, ActionRequest, or RenderRequest. You can access them using getAttribute() and setAttribute() on the appropriate request object. Objects stored in the request object are available for the duration of a single request. If you want to persist a value for subsequent requests, use the context object as described in the next section.

You can access the underlying request object from the EbiRequest interface, which provides the wrapper objects for each request type.

📖 For more information, see EbiRequest in the API help system.

Temporary values

You can also access request attributes as temporary values using get/setTemporaryValue() on EbiContext. The lifetime of the temporary value is the same as the lifetime of the EbiContext object.

For example, this code gets a request parameter and uses it to set a temporary value. The key is stored in the constant USER_CHOICE:

```
String userChoice =
   context.getEbiRequest().getParameter(USER_CHOICE);
if (userChoice != null)
   context.setTemporaryValue(USER_CHOICE, userChoice);
```

This method gets the value:

```
String choice = (String) context.getTemporaryValue(USER_CHOICE);
```

Session-level caching

This section describes ways to cache session-level data.

Using the Cache Manager

You can use the Cache Manager and session cache holder to cache nonserializable or serializable session objects in memory or on disk. You can cache data in either the object cache or the memory and disk integrated cache by using the appropriate methods on EbiSession.

📖 For more information, see "About the Cache Manager" on page 153 and "About the cache holder" on page 154.

Using the whiteboard

The whiteboard is a session-level cache for storing serializable objects. The whiteboard can be used to access commonly used values and for session-level failover. You can access the whiteboard using these methods on EbiContext:

Whiteboard access method	Usage
get/setValue()	Get a whiteboard value by specifying the key. Set a value by specifying a key and the object.
removeValue()	Remove a specified value from the whiteboard.
removeAllValues()	Remove all current values from the whiteboard.
getAttributeNames()	Enumerate the whiteboard key currently used.

session-level failover Session values that you want to preserve for session-level fail over must be cached on the whiteboard. Session-level failover refers to the ability of an application to retain temporary user data (state) across server failures in a cluster. The data is stored in a persistent storage repository (such as a database or file system shared by the servers in the cluster) so that it can be recovered by any server in the cluster in the event of a server failure.

IMPORTANT: Make sure the objects cached in the whiteboard are implementing java.io.Serializable. Otherwise, they will not be recovered if the session fails.

Each application server has its own level of support for session-level failover. Refer to your application server documentation for details.

 For information regarding session-level failover using the exteNd Application Server, see the chapter on server implementation notes in Application Server *Facilities Guide*.

Portlet session scopes

Portlet application data can be cached in a PortletSession object or in a PortletContext object, as defined in the javax.Portlet specification.

Portlet session attributes

The PortletSession interface defines two scopes for caching objects:

- APPLICATION_SCOPE Any object stored in the session using this scope is available to any other portlet that belongs to the same portlet application that handles a request identified as being a port of the same session.
- PORTLET_SCOPE Objects stored in the session using this scope must be available to the portlet during requests for the same portlet window.

You can access values in either scope using setAttribute() and getAttribute() on the PortletSession object by passing in the scope.

 For more information, see the API documentation for PortletSession.

NOTE: You can also access these scopes using the exteNd Director scoped paths feature. For more information, see Chapter 12, “Working with Scoped Paths and XPaths”.

Portlet context attributes

Attributes cached in the context are global for all users and all Web components in the portlet application. You can access these values using setAttribute() and getAttribute() on the PortletContext object. For more information, see the API documentation for PortletContext.

server-lifetime caching

Objects stored in the server-lifetime cache can be cached for enhancing performance and can be synchronized with other servers in a server cluster environment. This function is handled jointly by the exteNd Director Cache Manager and Cache Coordinator.

About the server-lifetime cache

You can cache objects in the server-lifetime cache by calling putObjectInCache() from the server-lifetime cache holder (EbiSrvLifetimeCacheHolder). The cached objects do not need to be implemented as javax.io.Serializable.

The server life-time cache is managed by the Cache Manager on each server. The cached objects are synchronized with the latest data stored in the database in a clustered environment. This function is handled by the *Cache Coordinator.*

📖 For more information, see Chapter 24, “Using the Cache Coordinator”.

Built-in cache holders

exteNd Director provides built-in server-lifetime cache holders for different type of subsystem runtime data, including:

- Content management artifacts
- Portal artifacts
- Directory users and groups
- Security settings
- Workflow processes

These caches are listed in the Director Administration Console under the **Cache Holders** section.

📖 For more information, see Chapter 22, “Using the General Configuration Section of the DAC”.

15 Logging Information

This chapter provides information about logging information in exteNd Director applications. It has the following sections:

About the exteNd Director logging facility

exteNd Director provides a runtime logging facility that writes information in one of several standard logs or in your own custom logs. You decide what level of detail you want in the logs. The output of the logs displays on the server console.

In addition to the standard logging facility, exteNd Director includes two logging providers that let you generate:

- Log information as well-formed XML files
- IPDR (Internet Protocol Detail Record) files

For more information, see Chapter 16, “Using the XML and IPDR Logging Providers”.

Uses for logging

In your application code, you can log all kinds of information. Uses for logging include:

- Recording errors and the application details when they occur
- Debugging during development
- Auditing application usage to determine what parts of your site are most successful
- Logging timing data to determine how long portions of the application take to run
- Logging transactions that are also stored in your database to provide an audit trail of sales or other agreements made with users

If you use analysis tools to study your Web site usage, you can write the log in a format expected by the tools.

What gets logged

Various classes in exteNd Director send status and information about errors to one of the standard logs. In addition, in your code you can instantiate a log variable for one of the standard logs or for your own log and write information to it.

Available logs

exteNd Director provides a log for each of the main packages. You can create additional logs in your application code when you get a log from EboLogFactory by specifying your own log identifier. The messages for all the logs are sent to the server console.

Package	Log identifier	Purpose of log: messages from the
com.sssw.cm	CM	Content Management subsystem
com.sssw.fw.directory	DIRECTORY	Directory subsystem
com.sssw.fw	FW	Framework subsystem
com.sssw.portal	PORTAL	Portal subsystem
com.sssw.re	RE	Rule subsystem
com.sssw.search	SEARCH	Search subsystem
com.sssw.fw.security	SECURITY	Security subsystem
com.sssw.fw.usermgr	USER	User subsystem
com.ssw.wf	WF	Workflow subsystem

Detail levels

exteNd Director supports four detail levels that determine how much information is written to the log. These detail levels are defined as constants on the EbiLog interface. Framework classes only send messages to the log that are at or below the current detail level. When you write the log you specify a detail level—and if the log's detail level is lower than the message being logged, the message does not appear.

Detail level name	Value	Description
DEACTIVATE_LEVEL	0	Nothing is written to the log
CRITICAL_ERROR_LEVEL	1	Critical errors are written to the log
ERROR_LEVEL	2	Unexpected errors are written to the log
WARNING_LEVEL	3	Expected errors are written to the log
INFO_LEVEL	4	Informational messages written to the log
TRACE_LEVEL	5	Debugging messages and messages about application progress are written to the log

You can change the detail level for a log in the DAC or in your exteNd Director application. For information about code that changes the level, see "Setting the detail level" on page 162.

Information in the log

The information in the logs has this format:

```
logname | detail-level | time | thread | message
```

Here the separator is a vertical bar. You can change the separator character. For example, if you want to export the log to a spreadsheet, you could create a tab-delimited or comma-delimited log.

Configuring the logs

In the DAC, you can specify the detail level for each of the logs, including your custom logs. You can also define your own custom logs by adding several entries to the config.xml for the target subsystem. For example, you could add a custom log called MyLog to the Portal Web tier by adding the following property settings to the config.xml for your application:

```
<property>
 <key>MyLog.LoggingLevel</key>
 <value>3</value>
</property>
<property>
 <key>MyLog.LogFieldSeparator</key>
 <value>|</value>
</property>
<property>
 <key>MyLog.LoggingProvider</key>
 <value>com.sssw.fw.log.EboStandardOutLoggingProvider</value>
</property>
```

You can also change log settings in code. For information, see com.sssw.fw.api.EbiLog and "Using logs in your application" next.

Using logs in your application

Some of the file generation wizards in exteNd Director include logging code. For example, when you create a new portlet using the Portlet Wizard, logging code is automatically added to the source code. The code gets a log object and writes messages at the trace level for each method. You can modify this code to get a different log. You can add logging statements that report exceptions that have been caught or that log information about the application's status. This section describes the code you would use.

Logging and scoped paths

You can use the exteNd Director scoped paths feature to access logs within pageflow and workflow applications.

For more information, see Chapter 12, "Working with Scoped Paths and XPaths".

Logging API

The Logging API includes these classes:

- com.sssw.fw.log.EboLogFactory—Use this class to get log objects
- com.sssw.fw.api.EbiLog—The log object, which has methods for getting and setting the log's detail level and writing messages in the log

Getting a log

To get a log object for one of the standard logs, use code like this:

```
import com.sssw.fw.log.*;
EbiLog log = EboLogFactory.getLog(EboLogFactory.PORTAL);
```

To create a unique log of your own, specify the log name as the argument:

```
import com.sssw.fw.log.*;
EbiLog log = EboLogFactory.getLog("WebAppLog");
```

It will use the default settings, which you can change with EbiLog methods:

- Detail level: CRITICAL_ERROR_LEVEL
- Field separator: vertical bar (|)

Setting the detail level

At any point in your code you can check or change the logging level for a log. Any changes affect future logging until the setting is changed again in code or in the DAC. For example, to set the logging level to trace, you would need to add this line of code:

```
log.setLoggingLevel(EbiLog.TRACE_LEVEL);
```

NOTE: The logging level must be set to trace if you want to see logging messages for a portlet that is generated by the Portlet Wizard, or any other wizard that generates logging code.

You can check the current level with isLevel() or one of the methods that check specific levels:

```
boolean traceOn = log.isLevel(EbiLog.TRACE_LEVEL);
boolean traceOn = log.isTrace();
```

For efficiency, it is useful to check the level before logging a message, described next.

Adding messages to the log

When you write code that sends a message to the log at a particular detail level, the message is logged only if the current detail is at that level or lower. If the detail level is CRITICAL_ERROR_LEVEL, when the application runs and calls trace() the message won't be logged.

Because messages won't always be logged, you can avoid the inefficiency of instantiating strings that won't be used by checking the detail level for logging the message.

For example, log.isCritical(), the lowest logging level, returns true for this logging level and higher, so critical messages are always written to the log unless logging is off.

log.isError() returns true for ERROR_LEVEL and higher, so error-level messages are logged unless the level is set to CRITICAL_ERROR_LEVEL.

To accept all logging, set the level to TRACE_LEVEL, the highest and most verbose logging level.

Logging messages

To log a message at a specific detail level, you can call the method for that level:

```
if (log.isTrace() )
{
   log.trace("Portlet returned content type: "
      + request.getContentType() );
}
```

In a catch block, you could log the error like this:

```
catch (EboException e)
{
   log.error("General portal exception:"
      + e.printStackTrace(context.getLocale()) );
}
```

You can also call the generic logString() method and specify the level:

```
log.logString("message", EbiLog.ERROR_LEVEL);
```

Broadcasting a message to all logs

You can send a message to all logs with the broadcast() method of EboLogFactory. You might use this when you want to set a mark in logs when conditions change so that you can note changes in application behavior:

```
EboLogFactory.broadcast("Application going into production now!");
```

The broadcast is sent at the critical error level so it will appear unless logging is deactivated. The message will appear once for each log.

Logging session information

You can identify individual sessions by including the user's ID as part of the log message. For example:

```
log.error(request.getUserPrincipal(getName() + log.getLogFieldSeparator()
   + "General portal exception: "
   + e.printStackTrace(context.getLocale()) );
```

Sample logging code for portlets

This section shows some sample logging code that could be used in a portlet application.

Logging for tracing and debugging

If you want to use logging to observe the execution of your portlet application, you might instantiate a PORTAL log like this:

```
EbiLog log = com.sssw.fw.log.EboLogFactory.getLog(
  com.sssw.fw.log.EboLogFactory.PORTLET);
```

The portlet code can generate content at the trace level and can log any caught exceptions at the error level. This example shows logging in the portlet doView() method:

```
public void doView( RenderRequest request, RenderResponse response )  throws
PortletException, java.io.IOException {

   try {
      PortletURL renderUrl = response.createRenderURL();
      renderUrl.setPortletMode( PortletMode.VIEW );

      response.setContentType(
          EbiPortletConstants.MIME_TYPE_HTML );
      PrintWriter writer = response.getWriter();

      // Build the screen of HTML, set it as the content
      StringBuffer sb = new StringBuffer();

      // Output code goes here
      sb.append( "View Mode" );
      sb.append( "<br></br>" );

     writer.print( sb.toString() );
   }

   catch (EboUnrecoverableSystemException e)
   {
      ... // code to respond to error
      // Error string replaces portlet content
      sb.replace(0, sb.length(),
         "msg describing what user should do");
      if (log.isCritical()) {
         log.criticalError(this.getPortletName()
```

```
            + " : Bad system error \n" + e.printStackTrace() );
      }
   }

   catch (EboApiException e)
   {
      ... // code to respond to error
      if (log.isError()) {
         log.error(this.getPortletName()
            + " : framework error \n" + e.printStackTrace() );
      }
   }
   catch (EboFactoryException e)
   {
      ... // code to respond to error
      if (log.isError()) {
         log.error(this.getPortletName()
            + " : factory exception \n" + e.printStackTrace() );
      }
   }
   catch (RuntimeException e)
   {
      ... // code to respond to error
      if (log.isWarning()) {
         log.warning(this.getPortletName()
            + " : runtime exception \n" + e.printStackTrace() );
      }
   }
    catch ( Throwable e ) {
      // Log other errors generated
         log.error(e);
         new PortletException( e );
       }
}
```

Documenting portal usage in a log

If you want to collect data about the clients your users use and the pages they access, you could add this information to an APP_USAGE log, as shown in this example:

```
// Instantiate the log
EbiLog log = com.sssw.fw.log.EboLogFactory.getLog(APP_USAGE);
// log data in this format:
// currentpage!browsername!browserversion!platform!callingpage

java.util.Map browserinfo = context.getBrowserInfo();
StringBuffer sb = new StringBuffer();

sb.append(context.getURI() );
sb.append("!");
sb.append(getMapValue(browserinfo,
          EboRequestHelper.BROWSER_NAME));
sb.append("!");
sb.append(getMapValue(browserinfo,
          EboRequestHelper.BROWSER_MAJOR_VER));
sb.append("!");
sb.append(getMapValue(browserinfo, EboRequestHelper.PLATFORM));
sb.append("!");
sb.append(context.getCallingPage() );

if (log.isInfo())
{
   log.info(sb.toString() );
}
// Private method to get browser information from the Map object:
private String getMapValue(Map map, String key)
```

```
{
   String val;
   if (map.containsKey(key))
   {
      val = (String) map.get(key);
   }
   else
   {
      val = "";
   }
   return val;
}
```

16 Using the XML and IPDR Logging Providers

This chapter includes instructions for using the XML and IPDR logging providers that ship with exteNd Director.

The chapter has the following sections:

About the XML and IPDR logging providers

In addition to the standard logging facility, exteNd Director includes two logging providers that let you generate:

- Log information as well-formed XML files
- IPDR (Internet Protocol Detail Record) files

IPDR defines an open, extensible record format for exchanging resource and service usage information. For more information about IPDR, see the IPDR.org Web site.

The behavior of the XML and IPDR logging providers is very similar to the behavior of the standard logging mechanism supported by exteNd Director.

For details on using the standard logging mechanism, see Chapter 15, “Logging Information”.

To use the XML or the IPDR logging providers, you first need to get the log object, which is an instance of EboLog. To do this you need to use the getLog() method on the EboLogFactory class, as shown below:

```
logXML = (EboLog) EboLogFactory.getLog( "XML Test" );
logIPDR = (EboLog) EboLogFactory.getLog( "IPDR Test" );
```

You can then add either of the two logging providers by using the addLoggingProvider() method on EboLog.

To add a logging provider object for XML logging, you might use this method call:

```
logXML.addLoggingProvider(EboUniqueXMLFileLoggingProvider.class.getName());
```

To add a logging provider object for IPDR logging, you might use this method call:

```
logIPDR.addLoggingProvider(EboUniqueIPDRFileLoggingProvider.class.getName());
```

Both providers have templates for the log file format, described next. Both kinds of templates should be placed in the **FrameworkService-conf** directory within the **FrameworkService.jar** file.

The name of the template can be specified as a tag in the log string. Each variable in the template should also be specified in the log string, unless it is one of the built-in properties:

```
logIPDR.audit("<template>sms_bin</template><WapID>aaaaa</WapID><pnummer>1</pnummer
>")
```

For more information on the built-in properties, see "Built-in properties" on page 169.

Working with XML templates

The template for an XML log file specifies one or more XML elements, as well as variables that will be replaced with log data. For example, the format for an XML file might be:

```
<?xml version="1.0"?>
<root>${event}</root>
```

When a new log file is created, the template is used to format the file. The ${} variable is replaced with the actual log strings. The value specified within the braces is also added as an enclosing element. For example:

```
<?xml version="1.0"?>
<root><event>logString2</event>
<event>logString2</event>
</root>
```

Working with IPDR templates

The template for an IPDR log file contains a set of standard IPDR elements. In addition, it may contain variables that will be replaced with log data. For example:

```
<IPDRDoc seqNum="6530" version="1.0">
 <IPDRRec info="Novell"/>
 <IPDR seqNum="${Sequence}" time="${Time}">
  <SS id="sms" service="bin">
   <SC>
    <v name="customer_id">${WapID}</v>
   </SC>
   <SE>
    <v name="service">sms-bin</v>
   </SE>
  </SS>
  <UE>
   <v name="pnummer">${pnummer}</v>
  </UE>
 </IPDR>
</IPDRDoc>
```

Built-in properties

The following built-in properties have been added to support XML and IPDR logging. Each of these properties can be set in the config.xml file for the Framework subsystem, or by calling a method on the EboLog class:

Property	Description	Value	EboLog method
logFileTemplate	The log file template to be used	The name of a template placed in the conf directory for the Framework Default value: **xml-file.tpl**	setLogFileTemplate (String logFileTemplate)
addLogDateInfo	Indicates whether a log Date should be added to the log string	true/false Default value: **true**	setAddLogDateInfo (String addLogDateInfo)
logDateFormat Mask	The mask to use for the log Date information	java.text.Simple DateFormat mask Default value: **dd/MM/yy HH:mm:ss**	setLogDateFormatM ask (String logDateFormatMask)
addLogTimeInfo	Indicates whether a log Time should be added to the log string	true/false Default value: **true**	setAddLogTimeInfo (String addLogTimeInfo)
logDateFormat Mask	Indicates whether a log Time should be added to the log string	java.text.Simple DateFormat mask Default value: **yyyy-MM-dd'T'HH:mm:ss'Z'**	setLogTimeFormat Mask (String logDateFormatMask)
addLogSequence Info	Indicates whether a unique Sequence should be added to the log string	true/false Default value: **true**	setAddLogSequence Info (String addLogSequenceInfo

Sample code

Here is sample code for using XML and IPDR logging:

```
// XML logging sample:
private com.sssw.fw.log.EboLog log;

log = (EboLog) EboLogFactory.getLog( "XML Test" );
log.addLoggingProvider(EboUniqueXMLFileLoggingProvider.class.getName()); log.audit("xml_file This is my
event");
log.removeLoggingProvider(EboUniqueXMLFileLoggingProvider.class.getName()); log2 =
  (EboLog) EboLogFactory.getLog( "IPDR Test" );

// IPDRF logging sample ...
private com.sssw.fw.log.EboLog log2;

log2.addLoggingProvider(EboUniqueIPDRFileLoggingProvider.class.getName());
log2.audit("sms_bin555123455");
log2.removeLoggingProvider(EboUniqueIPDRFileLoggingProvider.class.getName());
```

17 Working with JSP pages

This chapter describes how to use the exteNd Director tag libraries with JavaServer Pages (JSP). It includes the following topics:

About JSP pages and the exteNd Director tag libraries

exteNd Director provides a set of custom tag libraries for use with JavaServer Pages (JSP). These tags wrap many of the most commonly used methods in the exteNd Director API.

Tag library JAR files The exteNd Director tag libraries are distributed in the following JAR files:

- ContentMgmtTag.jar
- FrameworkTag.jar
- PortalTag.jar
- RuleTag.jar
- WorkflowTag.jar

For complete details on using tags in this file	See
ContentMgmtTag.jar	The tag library reference in the *Content Management Guide*
FrameworkTag.jar	The tag library reference in the *User Management Guide*
PortalTag.jar	The tag library reference in the *Portal Guide*
RuleTag.jar	The tag library reference in the *Rules Guide*
WorkflowTag.jar	The tag library reference in the *Workflow Guide*

TLD files Each of these JAR files has an associated tag library descriptor (TLD) file. The TLD file is an XML file that describes the tags in the library.

The exteNd Director tag libraries are described in the following TLD files:

- ContentMgmtTag.tld
- FrameworkTag.tld
- PortalTag.tld
- RuleTag.tld
- WorkflowTag.tld

To use a custom tag library, you need to include the tag library JAR file and the associated TLD file in the WAR that contains your JSP pages.

Where to put your JSP pages You should package your JSP pages in the WAR for your Web application, along with your exteNd Director resources. But the JSP pages should be at the root of the WAR, or in a subdirectory of the root. They should not be stored inside the resource set, or anywhere inside the WEB-INF/lib directory.

 For complete details on JSP and the Java Servlet API, see the Sun documentation.

Adding the JAR and TLD files to your project

When you use the Director Project Wizard to create your WAR file, you can select the tag libraries you need for your application. The JAR and TLD files you need are automatically added to the project:

- The **tag library JAR files** are placed in the **WEB-INF/lib** directory of the WAR file.
- The **TLD files** are placed in the **WEB-INF/tag** directory of the WAR file.

The web.xml file for the WAR file maps the tag library URIs to the included TLD files, as shown below:

```
<web-app>
...

 <taglib>
  <taglib-uri>/cm</taglib-uri>
  <taglib-location>/WEB-INF/tag/ContentMgmtTag.tld</taglib-location>
 </taglib>
 <taglib>
  <taglib-uri>/fw</taglib-uri>
  <taglib-location>/WEB-INF/tag/FrameworkTag.tld</taglib-location>
 </taglib>
 <taglib>
  <taglib-uri>/portal</taglib-uri>
  <taglib-location>/WEB-INF/tag/PortalTag.tld</taglib-location>
 </taglib>
 <taglib>
  <taglib-uri>/re</taglib-uri>
  <taglib-location>/WEB-INF/tag/RuleTag.tld</taglib-location>
 </taglib>
 <taglib>
  <taglib-uri>/wf</taglib-uri>
  <taglib-location>/WEB-INF/tag/WorkflowTag.tld</taglib-location>
 </taglib>

...
</web-app>
```

Using a custom tag in a JSP page

➢ **To use a custom tag in a JSP page:**

1 Add a tag library directive to the page to notify the server that the page relies on a custom tag library. Once you have added this directive to the page, all tags in the library are available to the page. Here's an example of a tag library directive that makes the tags described in the PortalTag.tld file available to the page:

```
<%@ taglib uri="/portal" prefix="ep" %>
```

This example assumes the web.xml file for the WAR file maps the URI to an appropriate TLD file. Here the URI **/portal** is mapped to **/WEB-INF/lib/PortalTag.tld**:

```
<web-app>
...
 <taglib>
  <taglib-uri>/portal</taglib-uri>
  <taglib-location>/WEB-INF/lib/PortalTag.tld</taglib-location>
 </taglib>

</web-app>
```

2 Add the custom tag to the page.

Here's an example that shows how to display a PID page on a JSP page:

```
<ep:displayPID PID="MyPID.html" />
```

18 Working with servlets

This chapter describes how to use servlets in an exteNd Director applications. It includes the following topics:

About servlets and exteNd Director applications

exteNd Director applications can include custom servlets. This version of exteNd Director supports servlets that conform to the Servlet 2.2 or 2.3 specification. Any servlets you create should conform to the specification supported by your target application server.

Where to put your servlets You should package your servlets in the WAR file that contains your Web application resources. Typically this WAR will be configured to use an internal resource set for exteNd Director resources. The servlets should be located in the WEB-INF/classes directory or in a JAR stored in the WEB-INF/lib directory within the WAR.

Using the exteNd Director API in a servlet

A custom servlet can access methods in the exteNd Director API. A servlet can use these methods to display portal pages and exteNd Director portlets, as well as to fire rules and access documents in the Content Management subsystem.

Example Suppose you want to use an exteNd Director portal page in a servlet. In this case, you would need to add some Java code that instructs the Portal Aggregation Controller to display the page. These are the basic steps you would need to follow:

1. Create a Portal context object based on the HttpServletRequest, HttpServletResponse, and Servlet Context objects.
2. Use the Portal Manager object to get an array of objects that provides information about pages defined to the Portal.

   NOTE: In the example below, the array of objects includes information about pages for all users. However, the API provides methods you can use to restrict the list by user.
3. Communicate with the Portal Aggregration Controller to render a portal response that includes the requested page.

Here is some sample Java code that shows how this is done:

```
package com.novell.afw.portal.aggregation;

import com.sssw.portal.factory.EboFactory;
import com.sssw.portal.api.EbiPortalContext;
import com.sssw.portal.api.EbiSharedPageInfo;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SampleServlet extends HttpServlet {

    public void init() throws ServletException {
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, java.io.IOException {
        callService(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, java.io.IOException {
        callService(request, response);
    }

    /**
     * Service a HTTP request
     */
    protected void callService(HttpServletRequest request, HttpServletResponse
response) throws ServletException, java.io.IOException {

        try {

            // Create a Portal Context based on the HttpServletRequest,
HttpServletResponse and Servlet Context
            // Portal context object is used to pass useful information around
            EbiPortalContext portalContext =
com.sssw.portal.factory.EboFactory.createPortalContext(request, response,
this.getServletContext());

            // Determine if a page was selected from the list of available pages
            String pageName = request.getParameter("pageName");

            if (pageName == null) {

                // No page name was supplied - show the user the available pages
(reports) they have access to
                StringBuffer pageBuffer = new StringBuffer();

                pageBuffer.append("<html>\n");
                pageBuffer.append("<head>\n");
                pageBuffer.append("<title>Available Reports</title>\n");
                pageBuffer.append("<link type=\"text/css\" rel=\"stylesheet\"
href=\"http://localhost:8080/Director/resource/portal-
theme/DottedBorder/theme.css\">\n");
                pageBuffer.append("</head>\n");
                pageBuffer.append("<body>\n");

                pageBuffer.append("<form id=\"reports\" method=\"post\"
action=\"\">\n");
                pageBuffer.append("  <div style=\"padding:5px;\">Available
Reports:</div>\n");
                pageBuffer.append("  <select name=\"pageName\"
style=\"height:150;width:290;\" size=\"10\">\n");
```

```
                // Get a list of shared pages from the Portal Manager
                EbiSharedPageInfo[] pageInfos =
EboFactory.getPortalManager(portalContext).getSharedPageInfoList(portalContext);

                for (int i=0; i<pageInfos.length;i++) {
                    pageBuffer.append("      <option>" +
pageInfos[i].getPageName() + "</option>\n");
                }

                pageBuffer.append("  </select>\n");
                pageBuffer.append("  <p/>\n");
                pageBuffer.append("  <input value=\"Show Report\"
type=\"submit\">\n");
                pageBuffer.append("</form>\n");
                pageBuffer.append("</body>\n");
                pageBuffer.append("</html>\n");

                response.setContentType("text/html");

                response.getWriter().print(pageBuffer.toString());

            } else {

                // A report (page) was selected from the list
                // Use the Portal's Aggregation Controller to display the selected
report (shared page)

EboPortalAggregationController.getController().renderPortalResponse(portalContext,
null, pageName);

            }

        } catch (Exception e) {
            e.printStackTrace();
            throw new ServletException(e);
        }
    }

    /**
     * the servlet is being taken out of service, clean up and free handles
     */
    public void destroy() {

    }
}
```

Here are the lines you would need to add to the web.xml to map the servlet to an URL pattern. This servlet listens on the URL pattern **/reports/***:

```
<servlet>
 <servlet-name>SampleServlet</servlet-name>
 <display-name>Sample Servlet</display-name>
 <description>Sample Servlet demostrating creating a custom servlet with hooks
into the portal.</description>
 <servlet-class>
  com.novell.afw.portal.aggregation.SampleServlet</servlet-class>
</servlet>

<servlet-mapping>
 <servlet-name>SampleServlet</servlet-name>
 <url-pattern>/reports/*</url-pattern>
</servlet-mapping>
```

19 Developing a Struts Application

This chapter describes developing Struts applications that take advantage of the services provided by exteNd Director. Topics include:

- About Struts
- Extending Struts with exteNd Director services
- How to implement Struts with exteNd Director services

About Struts

Struts is part of the Jakarta Project of the Apache Software Foundation. It is a framework that implements the **Model-View-Controller** (MVC) design pattern for J2EE Web applications (WARs).

Understanding MVC

MVC prescribes a way of partitioning application code to keep the user interface (the *view*) isolated from the business logic (the *model*). A *controller* determines how user requests are routed to pages and what business logic is invoked to process each request.

The combination of JSP pages for the view and servlets for the controller is known as **Model 2**. This is the currently accepted way to implement an MVC architecture in a Web application.

To learn more about MVC architecture, see the Sun J2EE Blueprints.

How Struts implements MVC

Struts is based on Model 2. It provides a servlet controller, tag libraries, and form classes that handle information display in JSP pages. It uses a configuration file to tell the controller what classes to instantiate to process application data.

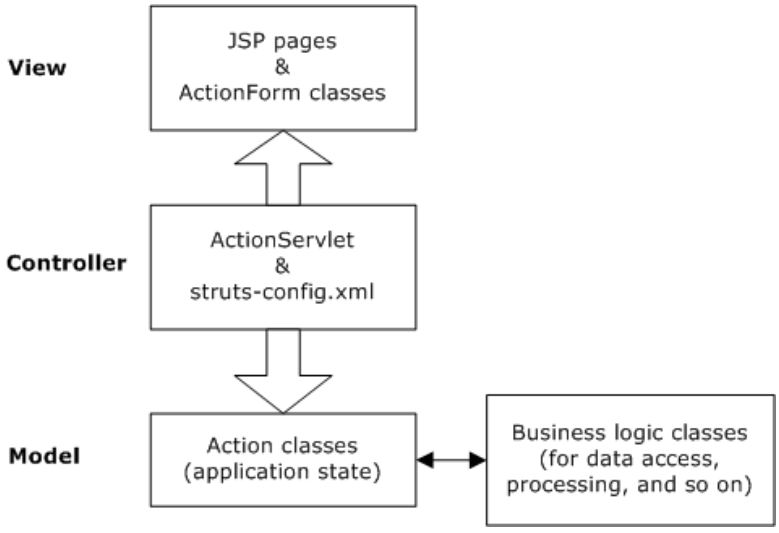
A typical Struts application includes:

- **JSP pages** with Struts custom tags that display text, create forms for data input, and process collections of data for presentation on the page
- **ActionForm bean classes** that populate forms with data and retain data for future requests
- **Action classes** that set up data for JSP pages and process user input
- **An action servlet** that acts as the controller, routing requests to action classes and selecting JSP pages to display
- **A configuration file** that defines the associations among URLs, action classes, form classes, and JSP pages
- **Resource files** that contain the text strings for the application, which can be provided in several languages

Here's a summary of how these pieces work together to implement MVC:

MVC part	Struts implementation
Model	Action classes use the request or the session to store application state information. They can instantiate business logic classes to handle application data. You write an action class for each URL the controller processes.
View	JSP pages and *ActionForm beans* display data and forms. ActionForm beans populate form fields with data and retain and validate that data. The data can remain available between requests, and the form can display the previously entered data again. You write an ActionForm class for each form on your JSP pages.
Controller	The ActionServlet class (or your extension of it) runs as a server process and processes URLs it recognizes. It reads the struts-config.xml file to find out what action classes to instantiate and what JSP pages to display for each URL. You can use the ActionServlet class as is or extend it to provide custom behavior.

The following diagram illustrates this architecture:



Example

Consider how a simple order entry system might be architected using Struts.

How it works The first component of the application is the order form where customers fill in order data. The HTML form itself is a JSP page that defines the text fields to be entered. This JSP page is mapped in the struts-config.xml file to an ActionForm class that collects and validates the input data. An Action class then processes the information and accepts the order. Finally, an action mapping tells the controller where to go next.

In this example, the form in the JSP page is defined using a Struts form tag:

```
<strutshtml:form action="order.do" name="orderForm" type="mine.OrderForm">
```

When a user clicks the Submit button, the controller knows to redirect to the mine.OrderForm class (which extends ActionForm) to collect the information entered. If there's a validation error, the controller can return to the form page; otherwise, it executes the order.do action.

Using the struts-config.xml file, the controller determines the specific Action class mapped to the order.do action and provides the collected order data to that class for processing. Depending on the result of the processing, the Action class returns an ActionMapping object to the controller that indicates what to do next. If the result is **success**, for instance, the controller looks in the struts-config.xml file to find the JSP page mapped to **success** for the Action class. In this application, the controller displays ThankYou.jsp if the order is successful and TryAgainLater.jsp if there is a problem.

Benefit Isolating the details of control flow in the struts-config.xml file means the application maintains flexibility, making it easier to add or change portions later. For example, this helps when adding a back-order module to the order entry application.

As mentioned earlier, a specific Action class processes the order information submitted by the user. That class may simply calculate the total cost of an order—or it may call to external code that validates the customer ID, processes the credit card data, and updates inventory before providing a response to the user. Because the Action class itself is part of the Struts framework, it's a good idea to implement application-specific business logic outside that class in reusable JavaBeans, EJBs, or other objects.

To learn more about Struts, visit jakarta.apache.org/struts/index.html.

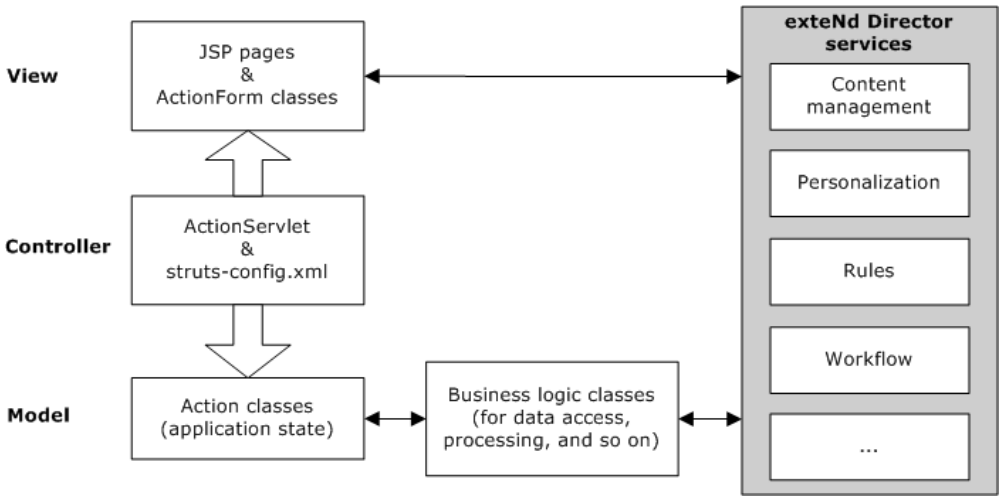
Extending Struts with exteNd Director services

By itself, the Struts framework can help you develop J2EE Web applications that follow a standard architecture (MVC, Model 2) and benefit from good design practices (such as the separation of presentation from business logic and the use of a configuration file for declaring details of control flow). Still, there are significant areas of application functionality that Struts does not address.

By adding exteNd Director services, you can extend the abilities of Struts applications in several specific areas, including:

- Business logic (via rules)
- Business process (via workflow)
- Dynamic content (via content management)

The following diagram illustrates the use of exteNd Director services with Struts:



Business logic

Struts limitation Even within a well-structured Struts application, business logic resides in coded objects that must be maintained and modified by programmers. This results in delays for business managers who need to make rapid changes to that logic in response to changing business plans and conditions.

exteNd Director benefit By using the exteNd Director **Rule** subsystem, you can maintain business logic in a much more accessible way, making it possible for both programmers and business managers to make changes. Once a rule change is made, it's immediately available to the application without the need for redeployment.

For example, suppose you want to give discounts to your best customers. You can create a rule called **isBestCustomer(custid)** that checks to see if a customer falls into the best category. Today a best customer is defined as having purchased $2000 worth of goods in one year, but maybe tomorrow a business manager will want to change that to $5000. Such modifications are easy—open the rule in the Rule Editor and change the amount from $2000 to $5000. As soon as you save the rule, the application starts using the $5000 requirement for best-customer discounts.

You can take advantage of rules in any of the logic components of a Struts application. In particular, you should consider using them to handle validation for ActionForm classes and business logic for Action classes.

Business process

Struts limitation Although Struts provides a good approach for handling control flow within an application, it doesn't go beyond the application to address how information and processes flow through a business. That's where the exteNd Director **Workflow** subsystem comes in.

exteNd Director benefit Workflow helps you map your business processes (including where individual applications fit) and move data through them. This eliminates the need to make particular applications aware of how they are linked, enabling you to develop them independently and keep them modular. For instance, an order entry application doesn't need to deal with what happens after an order is placed. It simply tells the Workflow subsystem that it has completed its part of the process.

Workflow manages the higher-level process logic. For example, it guides a customer's order from the order entry application to a person in shipping, then to someone in billing, and finally to a service rep who follows up with the customer to make sure everything arrived safely.

Dynamic content

Struts limitation Another important application requirement not addressed by Struts is support for dynamic content. For instance, while the use of JSP pages in Struts does help isolate the presentation layer of an application, there generally remains hardcoded HTML. In this scenario, content changes may require you to recompile.

exteNd Director benefit With the exteNd Director **Content Management** subsystem, content can be maintained and modified outside the application framework and fed into the application dynamically. Content Management includes features to version content, control access to content, control distribution of content, control expiration of content, and handle diverse styling of content via XML/XSL.

Suppose the main JSP page of an order entry application needs to display information about specials. Because specials are always changing, this content should be maintained dynamically; so you create a document in Content Management that holds the HTML for it. The JSP page is coded to obtain the content at runtime.

Nonprogrammers can use the **CMS Administration Console** to maintain and edit the content. You can put the content through an approval process and set a date for a new version of it to appear on the site. A rollback feature is available for cases where invalid content is created.

You can take advantage of styling capabilities to display the same content to different users in different formats. Content Management allows multiple XSL style sheets to be associated with a single document, so you don't need to maintain extra copies of content to support various looks.

How to implement Struts with exteNd Director services

To access exteNd Director services in a Struts application, you use several classes provided in the **StrutsPlus.jar** file, which is installed in the exteNd Director **utilities\Struts** directory. All of the classes you need are in the exteNd Director package **com.sssw.portal.struts**.

The com.sssw.portal.struts package has these classes:

- **DirectorAction**

  This class extends the Struts Action class. It provides several methods you can call to access common exteNd Director managers and delegates.

  To use DirectorAction, first extend the class. Then, in your DirectorAction subclass, implement the following abstract methods:

```
public abstract ActionForward perform(com.sssw.fw.api.EbiContext
   fwContext,com.sssw.portal.struts.EboStrutsContext    strutsContext )throws
java.io.IOException,    javax.servlet.ServletException;
public abstract void
   init(com.sssw.fw.api.EbiContextfwContext,com.sssw.portal.
   struts.EboStrutsContext strutsContext);
```

- **DirectorActionForm**

  This class extends the Struts ActionForm class. It provides several methods you can call to access common exteNd Director managers and delegates.

  To use DirectorActionForm, first extend the class. Then, in your DirectorActionForm subclass, implement the following abstract methods:

```
public abstract ActionErrors validate(com.sssw.fw.api.EbiContext
   fwContext,com.sssw.portal.struts.EboStrutsContext    strutsContext );
public abstract void reset( com.sssw.fw.api.EbiContextfwContext,
   com.sssw.portal.struts.EboStrutsContext strutsContext );
public abstract void
   init(com.sssw.fw.api.EbiContextfwContext,com.sssw.portal.
   struts.EboStrutsContextstrutsContext);
```

- **DirectorHelper**

  This is a utility class that provides several methods you can call to access common exteNd Director managers and delegates. These are the same methods as in DirectorAction and DirectorActionForm.

  To use DirectorHelper, you instantiate it, passing in the request and response as well as the servlet context. Then you can access the exteNd Director objects you need by calling the DirectorHelper get methods.

  DirectorHelper can be used anywhere, as long as you can supply the request, response, and servlet context.

The DirectorAction, DirectorActionForm, and DirectorHelper classes provide several get methods you can use to access common managers and delegates. You need to uncomment the get methods for the subsystems you plan to use in your application. To access a particular manager or delegate, you can then call the appropriate get method anywhere in your code. In your DirectorAction and DirectorActionForm subclasses, you would typically want to do this in the init method.

For example, to use the Rule subsystem, you would need to uncomment the following block of code:

```
/*
 //uncomment this section if you are using the rule subsystem
 public com.sssw.re.api.EbiRuleManager getRuleManager() throws
   com.sssw.fw.exception.EboFactoryException {
   try{
     return com.sssw.re.factory.EboFactory.createRuleManager();
   } catch (com.sssw.fw.exception.EboFactoryException ex){
     throw ex;
   }
 }
*/
```

To access the rule manager, you would then call the getRuleManager() method, as shown below:

```
public class MyDirectorAction extends DirectorActionForm {
...
 com.sssw.re.api.EbiRuleManager myRuleManager;
...
 public void init(com.sssw.fw.api.EbiContext
  fwContext,com.sssw.portal.struts.EboStrutsContext strutsContext)
 {
...
 myRuleManager = getRuleManager();
...
 }
```

V Deploying Applications

Explains how to deploy an exteNd Director application

20 Deploying exteNd Director Applications

This chapter provides information about deploying exteNd Director applications. It contains these sections:

Deploying an exteNd Director project

After you build and archive your project, you can deploy it to a J2EE application server. You can deploy a newly created project—you do not have to add any application-specific functionality to a project before deploying it.

The following table lists the deployment configurations supported for each exteNd Director project type:

exteNd Director project	Shared library configuration	Deployment configuration description
Portlet application	Full	The target application server must: ◆ Be configured to use full shared libraries ◆ Have an exteNd Director portal already deployed
	3rd party only	Must be incorporated in an exteNd Director EAR or WAR
	Nonshared	Must be incorporated in an exteNd Director EAR or WAR
exteNd Director project	Full	The target application server must: ◆ Use shared libraries ◆ Not have a portal deployed
	3rd party only	The target application server must: ◆ Be configured as a 3rd party JARs only server Each exteNd Director project must use its own database
	Nonshared	The target application server must: ◆ Be configured as a nonshared library server Each exteNd Director project must use its own database

📖 For more information on shared library configurations, see "Changing a project's shared library configuration" on page 53.

Depending on the server you are deploying to, you'll need to do some predeployment setup and possibly some post-deployment configuration.

The deployment process includes these tasks:

- Predeployment tasks
- Deployment tasks
- Post-deployment tasks

Predeployment tasks

The tasks in the following table apply to Portlet Application Projects and Director projects. Before you deploy your application, make sure of the following:

Make sure that	For information see
You have a relational database available to the application server you are deploying to.	"Setting up an exteNd Director database" on page 189
For production deployments, you have: ◆ Turned off vulturing ◆ Generated a new AES key	◆ For information on vulturing, see "Dynamic loading of resources and classes" on page 77 ◆ To generate a new AES key, see "Changing the configuration" on page 49
Your project and the target server use the same shared library configuration	"Changing a project's shared library configuration" on page 53
The necessary JARs and files are copied to the correct location for your server	"Setting up JARs and the server's classpath" on page 189
For portlet applications, you've determined if you can use asynchronous rendering	The section on asynchronous portlet rendering in the *Portal Guide*.
Your server's predeployment tasks are complete	◆ "Tomcat predeployment tasks" on page 191 ◆ "LDAP predeployment tasks (eDirectory only)" on page 191

Setting up an exteNd Director database

Many of the exteNd Director subsystems require a relational database to store persistent data. Follow the steps outlined below to set up a database for use by exteNd Director:

Step	Task	For more information
1	Use your DBMS tools to create a database for the exteNd Director tables.	For more information on creating a database, see your DBMS documentation. For more information about the supported databases, see the *Release Notes*.
2	Use your server's tools to create a connection pool for the database you created in Step 1.	See your application server's documentation for creating a connection pool. For more information about the supported application servers, see the *Release Notes*. For help on configuring Tomcat, see "Tomcat predeployment tasks" on page 191. **IMPORTANT:** To use an Oracle database with the Novell exteNd Application Server, you must use the application server's Add Database functionality (not the connection pool).
3	Make sure the database driver you are using to connect to the exteNd Director database is in the application server's classpath	For information on adding to the server's classpath, see your application server's documentation.

 For a list of the exteNd Director database tables, see "About exteNd Director database tables" on page 204.

Setting up JARs and the server's classpath

exteNd Director requires you to use exteNd Director's version of these JARs:

- xercesImpl
- xalan
- Phaos_Crypto_FIPS
- Phaos_SSLava
- Phaos_Security_Engine

You'll need to copy the file from the exteNd Director install to a specific directory on your application server.

1 Copy each of these files from this install location:

Copy this JAR	From this exteNd Director install directory
xalan.jar	extend5\Director\templates\Director\library
xercesImpl.jar	
Phaos_Crypto_FIPS	Common/jre/lib/ext
Phaos_SSLava	
Phaos_Security_Engine	

2 Place the copy of exteNd Director's version of these JARs in the proper location for your application server as described in the next table.

NOTE: If you are using the exteNd Application Server, these files are installed into the proper location by default, so you are not required to make any changes.

IMPORTANT: Restart the server after copying these files.

Server	JAR	Instructions
BEA WebLogic	xalan	**1** Create a subdirectory called **endorsed** in the JRE used by WebLogic. For example: `c:\bea\jdk141_02\jre\lib\endorsed` **2** Copy exteNd Director's xalan.jar to the endorsed directory.
	xercesImpl	**1** Copy to the server's \lib directory. **2** Edit the server's start command to reference the xercesImpl.jar as the first item in the server's Classpath.
	Phaos_Crypto_FIPS Phaos_SSLava Phaos_Security_Engine	◆ Copy to the server's jre\lib\ext directory.
IBM WebSphere	xalan	**1** Copy to the server's \lib directory. (You should first save WebSphere's copy of xalan.jar to a different location.)
	xercesImpl	**1** Rename WebSphere's xerces.jar and move it out of the WebSphere\lib directory. **2** Copy exteNd Director's xercesImpl.jar to WebSphere's \lib directory.
	Phaos_Crypto_FIPS Phaos_SSLava Phaos_Security_Engine	◆ Copy to the server's jre\lib\ext directory.
Apache Tomcat	xalan	◆ Copy to server's \common\endorsed directory.
	xercesImpl	◆ Copy to server's \common\endorsed directory.
	Phaos_Crypto_FIPS Phaos_SSLava Phaos_Security_Engine	◆ Copy to the server's jre\lib\ext directory.

IMPORTANT: If you are developing on a Windows platform and deploying to a non-exteNd application server running on UNIX, copy the Phaos_Crypto_FIPS_UNIX.jar from the Common\lib\other directory to the directory for your server specified above. Rename it Phaos_Crypto_FIPS.jar.

Tomcat predeployment tasks

1 Locate the FrameworkService-conf and DirectoryService-conf files for your project type:

Archive type	Project location
EAR	\Library\ConfigService\ConfigService.spf\
WAR	WEB-INF\lib\ConfigService\ConfigService.spf\

2 Make sure the following Framework services settings are correct:

In the **FrameworkService-conf** config.xml file:

Key	Value
com.sssw.userTransaction.enable	False (unless you have a JTA extension installed)
com.sssw.fw.datasource.jndi-name	Matches the name of the connection pools that you created to store your exteNd Director tables

In the **DirectoryService-conf** config.xml file. The values for PersistManager are:

Key	Value
DirectoryService/realms/readable	com.sssw.fw.directory.api.EbiPersistMgrRealm
DirectoryService/realms/writeable	com.sssw.fw.directory.api.EbiPersistMgrRealm

The values for LDAP are:

Key	Value
DirectoryService/realms/readable	com.sssw.fw.directory.api.EbiJndiLdapRealm
DirectoryService/realms/writeable	com.sssw.fw.directory.api.EbiJndiLdapRealm

TIP: Using the PersistManager realm causes the application database to function as the user authentication repository. The LDAP realm specifies that user authentication repository is an eDirectory server.

LDAP predeployment tasks (eDirectory only)

If you are using an eDirectory LDAP realm configuration, you need to:

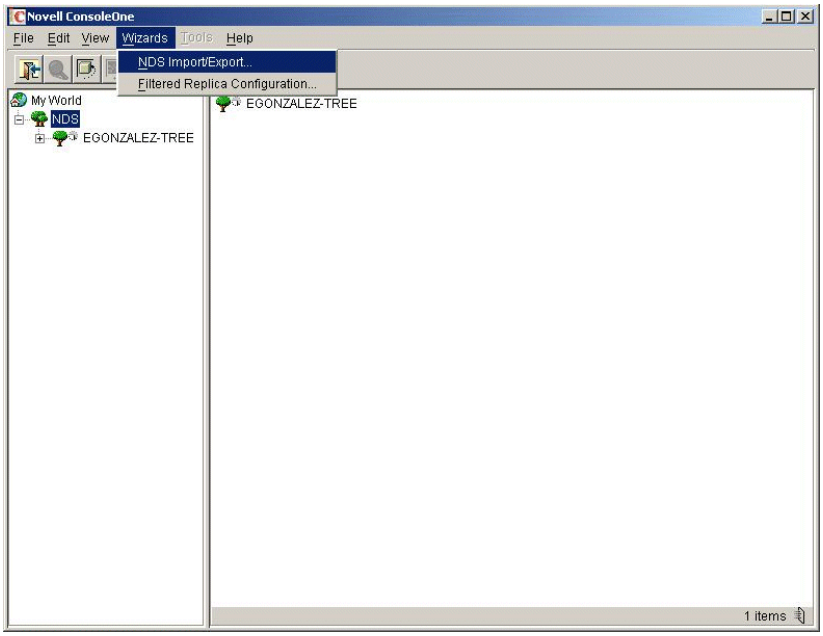
- Import a UUID auxiliary class
- (Optional) Set up your LDAP server to use SSL

Importing the UUID auxiliary class Before you deploy a project that implements one of the LDAP application server realms, you need to import the provided auxiliary class specified in the UUID auxiliary class property in the Directory LDAP Configuration panel. exteNd Director uses this class to access portal users in the LDAP tree.

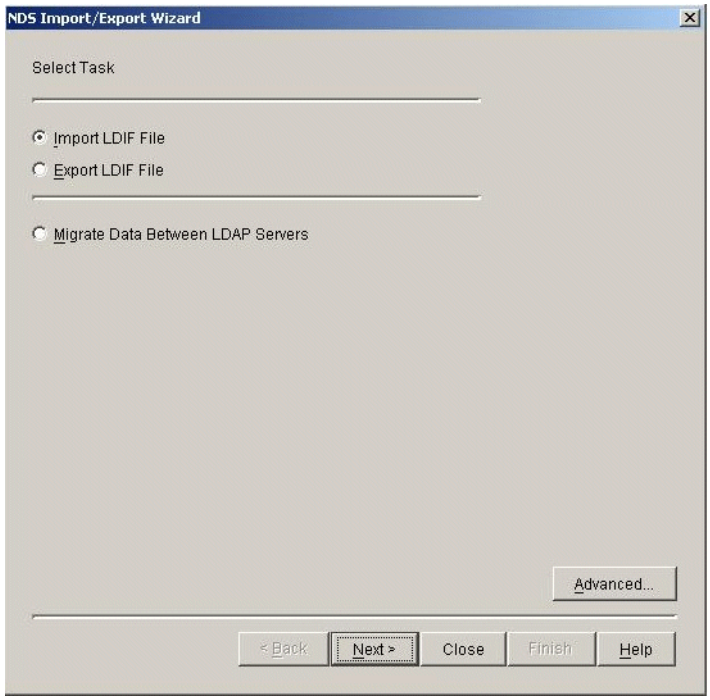
You import this class using the NDS Import Wizard in the Novell ConsoleOne® eDirectory tool.

➢ **To import the UUID auxiliary class in ConsoleOne:**

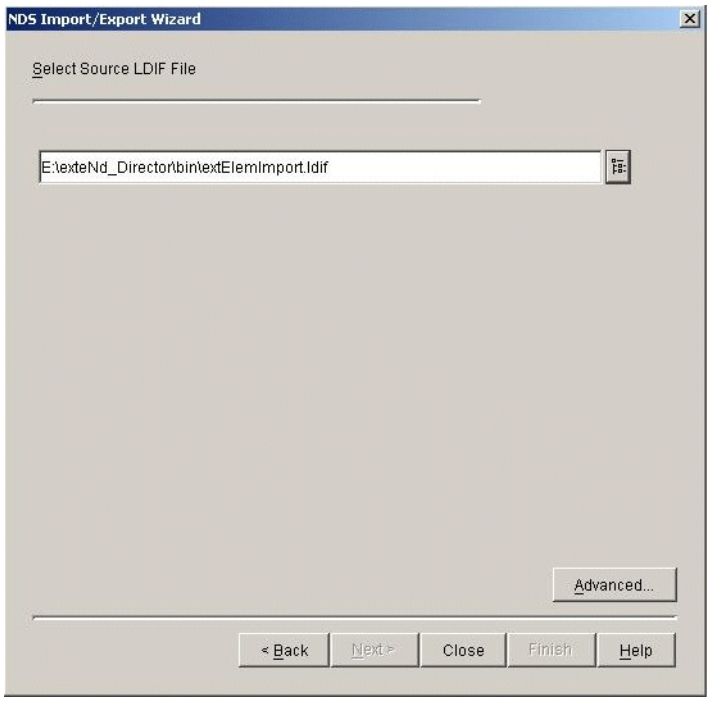
1 With the **NDS** container selected in ConsoleOne, select **Wizards>NDS> Import/Export**:



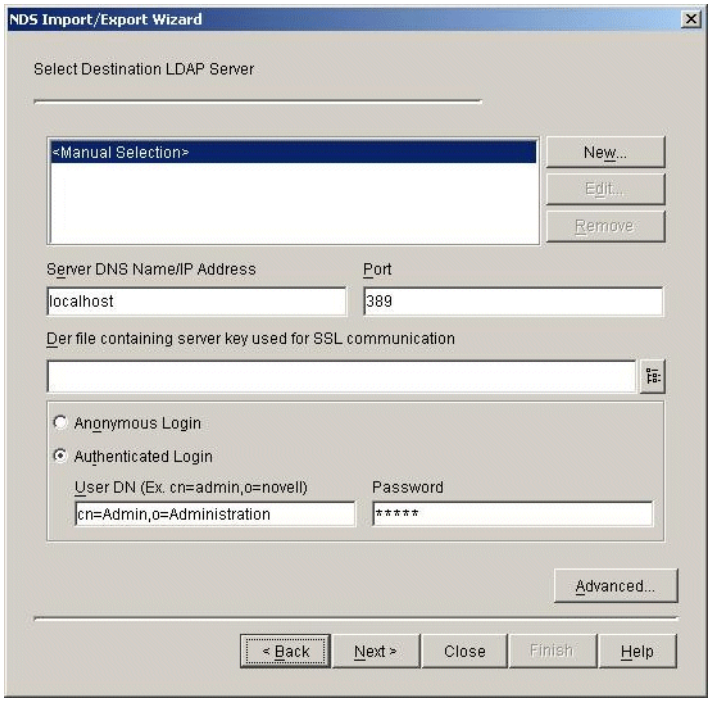
2 Click **Import LDIF File** and choose **Next**:



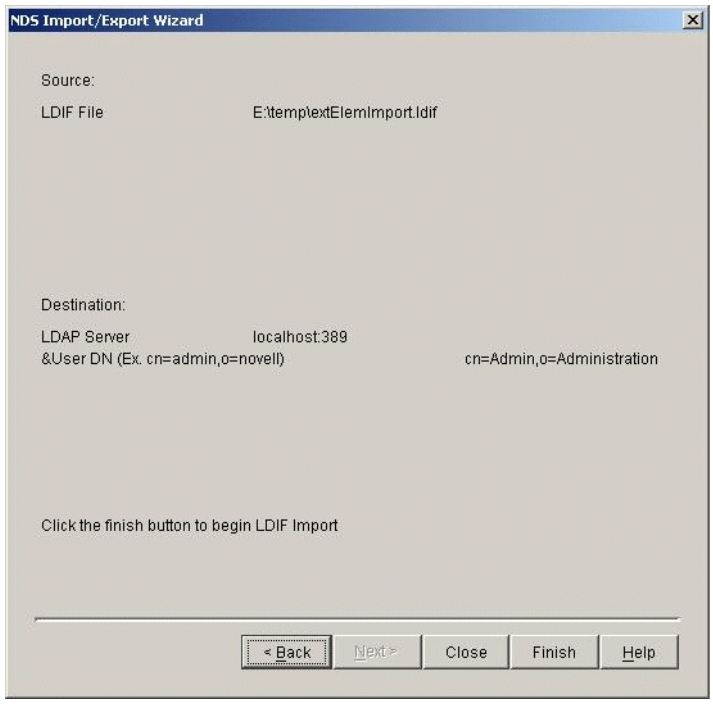
3 Navigate to the **ldif** file in your exteNd Director installation path and select it (it is located at bin/extElemImport.ldif). Click **Next**:



4 Verify the LDAP host name and port, choose **Authenticated Login**, and specify your administrator DN (distinguished name) and password:



5 Verify the information and click **Finish**:



Configuring and using SSL for LDAP connections Complete the procedures described here if you want to connect to your LDAP realm using Secure Socket Layer (SSL).

NOTE: SSL connections are slower than plain or clear-text connections. The initial portal connection can take up to 30 seconds to be established.

➢ **To configure the LDAP server to support SSL:**

1 In ConsoleOne, open the properties on the LDAP Server object that represents the LDAP server you are using with exteNd Director.

2 Select the **SSL Configuration** tab.

3 In the SSL Certificate field, select an **SSL Certificate** object.

4 Make note of the SSL Port (typically 636).

5 Make sure the **Disable SSL Port** option is **not** checked.

6 Save the settings and refresh the NLDAP server:

6a Open the properties of the LDAP server.

6b Click the **Refresh NLDAP Server Now** button on the **General** page.

➢ **To export the Trusted Root Certificate:**

1 Open the properties on the SSL certificate object that you configured in the preceding procedure.

2 Select the **Certificates** tab.

3 Select the **Trusted Root Certificate** subtab.

4 Click **Export** and save the file in **binary DER format**, typically named **TrustedRootCert.der**.

➢ **To download and install the Java Secure Socket Extension (JSSE):**

This step is required only if your server is running with a JRE older than 1.4 and is not already configured to use JSSE.

Follow the installation instructions for JSSE—summarized as follows:

1 Copy JSSE.JAR, JNET.JAR, and JCERT.JAR to the server's JRE extensions directory (for example: jre/lib/ext).

2 Find and edit the **java.security** file, located in the lib/security directory of the server's JRE (for example: jre/lib/security/java.security).

3 Follow the directions at the beginning of the document to add the JSSE SSL provider:

- Add a line to the security providers section using the format below, replacing *name* with the next provider number in succession:

```
security.provider.name =com.sun.net.ssl.internal.ssl.Provider
```

➢ **To import the Trusted Root Certificate into your cacerts or jssecacerts trust store file:**

1 Find the **cacerts** or **jssecacerts** file. It is located in the lib/security directory of the server's JRE (for example: jre/lib/security/cacerts).

2 Find **keytool**, located in the /bin folder relative to your Java home folder.

IMPORTANT: You must use a keytool that comes with JVM 1.3 or later.

3 Run the following command, making replacements listed below:

```
keytool -import -alias aliasName -file TrustedRootCert.der -keystore cacerts -
storepass changeit
```

- Replace *aliasName* with a unique name for this certificate.
- Make sure the full path for cacerts and the full path for TrustedRootCert.der are specified.
- Note that **changeit** is the default keystore password. Use the appropriate keystore password if it has been changed.

➢ **To configure exteNd Director to use SSL in the Directory service:**

1 Open your exteNd Director project.

2 Select **Project>Director Project>Configuration**.

3 Click the **Directory** tab and then click the **Directory Ldap Options** lower tab.

4 Change the LDAP host to include the SSL port for eDirectory (for example: localhost:636).

5 Make sure **Use SSL** is checked and click **OK**.

6 Rebuild your project and redeploy.

Deployment tasks

Once you've completed all the predeployment steps, you can start deployment.

- To deploy to IBM WebSphere Advanced Edition, you must use the WebSphere deployment tools; see "Using IBM WebSphere deployment tools" on page 196.
- To deploy to any other supported server, see "Using exteNd Director deployment tools" next.

Using exteNd Director deployment tools

To use the exteNd Director deployment tools, complete these deployment tasks:

Step	Task	For more information
1	Define a server profile	See the chapter on creating a server profile in *Utility Tools*
2	Define deployment settings	See the chapter on creating deployment settings in *Utility Tools.*

Step	Task	For more information
3	Create a deployment document (if required by the target server)	See the chapter on Deployment Plan Editor in *Utility Tools*.
4	Build and archive the project	See the chapter on Projects and Archives in *Utility Tools*.
5	Deploy the project	See the chapter on deploying an exteNd Director project in *Utility Tools*. **IMPORTANT:** When you deploy multiple WARs to the Apache Tomcat server, you must deploy the WAR containing the exteNd Director Portal **first**, verify that the exteNd Director tables are created, then deploy any remaining portlet application WARs.
6	Move portal data (such as container, shared, personal pages, and portlets) from your test environment to the production environment.	See the chapter on moving portal data in the *Portal Guide*.

Using IBM WebSphere deployment tools

To deploy to WebSphere Advanced Edition, you must use the WebSphere Advanced Administrative Console—you cannot use the exteNd Director deployment tools. Here are the general steps to follow when using the console. For more details, see the IBM documentation.

Step	Task	Details
1	Build the EAR in exteNd Director.	Use exteNd Director's Build and Archive or Rebuild All and Archive commands; both commands are described in the section on compiling, building, and archiving in *Utility Tools*.
2	Use the Administrative Console to deploy and start the application.	Use your DBMS tools to verify that exteNd Director tables are in the database. The list of tables that should be added are included in "About exteNd Director database tables" on page 204.
3	Set ClassLoader Mode and ClassLoader Visibility	After successfully deploying your application, make sure: ◆ ClassLoader Mode is set to PARENT_FIRST ◆ ClassLoader Policy is set to Application

Post-deployment tasks

After deployment you need to:

Step	Task	For more information
1	Do the post-deployment tasks that apply to the application server you deployed to	See the section specific to your server: ◆ "Novell exteNd Application Server" on page 197 ◆ "IBM WebSphere" on page 197 **NOTE:** For the Tomcat and BEA WebLogic servers, no action is required.
2	Make sure the Locksmith user is a valid user in the realm	—

Step	Task	For more information
3	Test the deployment*	See "Testing the deployment" on page 199
4	(Optional) To enable contextual searching, configure Autonomy	See the section on configuring your environment for conceptual searching in the *Content Search Guide*.

* If you redeployed on a shared library server, and also copied new or updated existing JARs in the shared library directory, you'll need to restart the server after the redeployment.

Novell exteNd Application Server

If you are using connection pools to access the exteNd Director database, you do not need to perform any post-deployment tasks—so you can skip this section.

If you used the deprecated AddDatabase feature to access the exteNd Director database, you'll need to perform the procedure that follows. (Because the deployment procedure adds tables to the exteNd Director database, the schema changes render the server's snapshot of the database schema out of sync. To fix this, you need to synchronize the database schema.)

➢ **To synchronize the database schema:**

1 While the server is running, start the **Server Management Console** (SMC).

2 Select the **Databases** panel.

3 In **Settings for database**, select your exteNd Director database.

4 Click **Synchronize Database Schema**.

IBM WebSphere

You'll need to do these postdeployment tasks only if your project does either of the following:

- Uses the WebSphere custom realm (at project creation, you chose WebSphere as your realm). Because this acts as a custom registry, you'll need to make the custom registry entries described in the following procedures.
- Uses an LDAP realm (at project creation, you chose LDAP as the realm). You'll just need to follow the procedure "To define global security:" on page 198.

➢ **To define general properties:**

1 In the Custom User Registry, define the following properties with the values shown:

Property	Value
Server User ID	admin
Server Password	admin
Custom Registry Classname	com.sssw.fw.server.websphere.realm.EboWebSphereRealm
Ignore case	Not selected

2 Click **Save**.

➢ **To define Custom User Registry Custom properties:**

1 Define these values:

Name	Value

driver	Enter the JDBC driver class name used to connect to the exteNd Director database. This is not the Datasource class name. For example, the JDBC driver class name for the Microsoft SQL Server type 4 driver is:
	`com.microsoft.jdbc.sqlserver.SQLServerDriver`
	For Oracle Thin Driver:
	`oracle.jdbc.OracleDriver`

url	The JDBC URL to connect to the exteNd Director database.
	For example, for Microsoft SQL Server type 4 driver:
	`jdbc:microsoft:sqlserver://host:port;user=myuser;password=secret;databasename=Directordatabase`
	For Oracle Thin Driver:
	`jdbc:oraclce:thin:@wd40:1521:ocl`

user	Enter the username that you used when you added that database to the server.
password	Enter the password.

2 Click **Save**.

➢ **To define global security:**

1 Define these values:

Property	Value
Enabled	Selected
Enforce Java 2 security	Not selected **NOTE:** If you want to use Java 2 security, change the value to **selected** and configure the Java security policy file.
Active User Registry	Custom or LDAP

2 Click **Save**.

3 Restart the server for all property settings to take effect.

Testing the deployment

➢ **To test the deployed application:**

1 In your browser, try the URLs that match your deployment configuration:

Server	URL

Novell exteNd—deployed to SilverMaster	`http://server:port/context/portal`
	For example:
	`http://localhost/ExpressPortal/portal`

BEA WebLogic	
IBM WebSphere	

Novell exteNd—deployed to non-SilverMaster	`http://server:port/database/context/portal`
	For example:
	`http://localhost/MyDatabase/MyPortal/portal`

Variables in the URLs In this procedure, the variables have these meanings:

- *server* is the name of your application server.
- *port* is the port number for your application server. The defaults are:

Server	Default port
Novell exteNd	80 (Windows) 8080 (UNIX) 83 (NetWare)
Tomcat	8080
WebLogic	7001
WebSphere	9080

- *database* is the database to which you deployed the archive (Novell exteNd only)
- *context* is the namespace for the exteNd Director application, which you specified in the Project Wizard

What happens to exteNd Director subsystems at deployment

At deployment, exteNd Director applications are compiled, archived, and deployed to the specified server (described in the deployment chapter of *Utility Tools*). For applications that rely on one or more exteNd Director subsystems, there are additional activities that occur (without user intervention) at deployment. They are described in the following sections:

- How the subsystems register themselves with the Framework
- How the subsystems access persistent data
- How the subsystems access application resources

How the subsystems register themselves with the Framework

To make its services available, a subsystem must register itself with the Framework. The process of self-registration involves loading configuration and service information into the Framework. Once this information has been loaded, the appropriate factories can produce the correct implementations for each requested service. The registration process is initiated by the **boot servlet**, which starts up automatically when you deploy an exteNd Director EAR or restart your application server. An autostart servlet must be the first servlet to load within an exteNd Director EAR.

Boot process During the framework boot process, the boot servlet starts up the base factory for the framework, which performs these operations:

1. Reads the **config.xml** file for each subsystem into the framework's memory space.

   The config.xml file sets the configuration properties for a subsystem. It is typically located in the *subsystem-name*-conf subdirectory of the ConfigService.spf.

2. Performs any database processing required for each subsystem. For each subsystem that needs to load data into a database, the base factory creates the schema and loads the data, if necessary.

   Each subsystem that requires database processing delivers scripts for creating the schema and loading the data. These scripts are located in the database subdirectory of the *subsystem-name*-conf subdirectory of the ConfigService.spf.

   📖 For more information about how the subsystems perform database processing, see "How the subsystems access persistent data" on page 200.

3. Reads the **services.xml** file for each subsystem into the framework's memory space.

   The services.xml file sets properties for each of the services associated with a subsystem. It is located in the *subsystem-name*-conf subdirectory of the ConfigService.spf.

4. Notifies subsystem-specific service loaders that the main processing is done.

5. Starts all autostart services listed in services.xml files.

   Autostart services have a startup property value of A (for automatic). Those services that have a startup element of M (for manual) are not started.

How the subsystems access persistent data

The following subsystems depend on a database to store persistent information:

- Content Management
- Directory
- Security
- User
- Workflow

After the Framework boot servlet reads the **config.xml** file for each subsystem, it performs any database processing required for the subsystems. The boot servlet checks the config.xml file to determine if it needs to create the schema and load data. It does this by examining the following configuration properties:

Property	Description
db-load-on-startup	Tells the boot servlet to check the database to see if its schema was created previously: ◆ If the schema has not yet been created, it creates the schema automatically. ◆ If the schema has been created, it does not perform any database processing. This test ensures that the process of uploading the schema does not occur every time the server is started (or every time the EAR is deployed).
test-db-on-startup	Tells the boot servlet which table should be used to test for schema creation.

For example, the config.xml file for the Directory subsystem has these settings for the db-load-on-startup and test-db-on-startup properties:

```
<property>
 <key>DirectoryService/db-load-on-startup</key>
 <value>true</value>
</property>
<property>
 <key>DirectoryService/test-db-on-startup</key>
 <value>AUTHGROUPS</value>
</property>
```

When database processing is required for a subsystem, the boot servlet executes the scripts in the database subdirectory of the *subsystem-name*-conf subdirectory of the ConfigService.spf.

NOTE: Do not edit the database scripts provided with the exteNd Director subsystems. These scripts should be used as they are.

How the subsystems access application resources

exteNd Director subsystems often require access to application resources that are not stored in a database or defined in the configuration and service elements for the subsystems. Resource sets provide a known location for these resources. A resource set holds definitions for rules, pageflows, portlets, styles, and various descriptors that implement features provided by exteNd Director subsystems. It can also hold Java classes and images for your application.

Resource sets are also a tool for streamlining development, because they can be configured to load resources from disk as well as from a deployed JAR. This dynamic loading of resources allows you to modify and test changes without having to redeploy the whole exteNd Director EAR.

For information on how to use resource sets, see Chapter 6, “Using the Resource Set in an exteNd Director Application”.

Troubleshooting the deployment

This section contains troubleshooting for the following categories:

- General troubleshooting
- Troubleshooting BEA WebLogic deployments

General troubleshooting

Problem	Cause	Solution

You encounter the following error:
`Server Console Trace:`
`----------`
`WARNING: This portal app context, Director51, does not match the portal.context property set in the PortalService-conf/config.xml file. Only one portal per data base is allowed. Data has been loaded using the previous portal context. To correct this you must revert back to the previous portal name of, null, please consult the documentation. java.lang.reflect.InvocationTargetException`
	Your application server is configured to use shared libraries and an exteNd Director portal is already deployed—you are attempting to deploy another application that includes a portal.	In a shared library environment, you are restricted to a single deployed portal. You have two options. You can:
		◆ Change the server and project configuration to nonshared library so that you can deploy multiple portals.
		📖 For more information, see “Changing a project’s shared library configuration” on page 53.
		OR
		◆ Complete the following steps (that will undeploy your already deployed portal so that you can deploy the new project.
		1 Use your server's tools to undeploy the archive containing the existing exteNd Director portal.
		2 Create a new exteNd Director database for the new portal, and create a new connection pool for it.
		3 Build the project for the new portal (and a deployment plan if needed).
		4 Choose **Project>Director>Shared Lib**.
		5 Click **Copy JARs** and copy the shared library JARs to the appropriate location for your server.
		NOTE: If you do not perform this step, the application server will still contain references to the previously deployed portal's configuration, and you'll get the same error.
		6 Deploy the archive.

Troubleshooting BEA WebLogic deployments

Problem	Action
Redeploying an EAR fails	If you’ve deployed before and the **update** deployment option fails to replace the existing EAR, you can use the WebLogic Console to remove the EAR. Then open the Deployment Settings, change the Deployment Options value back to **deploy**, and try deploying again.
You did not generate targets	After deploying, you may receive an error stating that no target was set. If so, you can use the WebLogic Console to specify the target server for each of the Web modules in the EAR.

Problem	Action

Out-of-memory errors occur	When deploying multiple times to WebLogic, you may see Out of Memory errors occur. This can occur when deploying two separate EARs. By default, WebLogic.cmd sets memory to 64, as shown below:
	`-ms64m -mx64m`
	To increase the amount of memory available, increase the value as shown below:
	`-ms64m -mx256m`

Portal URLs are not relative	After deploying, you may notice that the some of the URLs associated with your application are not relative, but instead reference localhost. To fix this problem, you may need to modify the configuration for the machine you set up for deployment. Select the machine and click the Server tab. Then, if necessary, add the server (for example, myserver) to the Chosen list and click Apply. Once you've made these changes, restart your server.
You are unable to upload large files	When you try to upload large files to WebLogic, you may find that the operation times out. This can happen if the JTA timeout setting in WebLogic is not too small. The default timeout setting is 30 seconds. To increase the timeout setting, select JTA under mydomain. Then increase the value in the Timeout Seconds field and click Apply.

TIP: Be sure to check the *Release Notes* for additional information about configuring WebLogic for exteNd Director.

Changing your deployment configuration

When you configure an exteNd Director application, you make choices based on the application server you plan to use. If you want to deploy the application to a different server later, there are several things to change. All changes can be made using exteNd Director editors.

For details on making changes, see the procedures in Chapter 3, "Reconfiguring exteNd Director Projects".

The items you need to change are as follows:

What to change	For information on how to change it (paths shown for archive layout in exteNd Director)
Realm for user authentication	See "Directory configuration" on page 38.
Locksmith ID	See "Framework configuration" on page 41.
JNDI name for the application database	
URIs that proxy resource sets use to redirect resource requests to remote resource sets	See "Working with entries for resourcePath and libPath" on page 84. **NOTE:** If you are also changing the server where the remote resource set is deployed, you will need to edit the proxy resource set's URI to point to the new location.

About exteNd Director database tables

The exteNd Director database tables hold information that the subsystems need to persist.

NOTE: The items listed are reserved for exteNd Director's use. This listing is provided for informational purposes only.

Subsystem	Table
Directory	AUTHGROUPBINDINGS
	AUTHGROUPS
	AUTHUSERS
Content Management	CMCATEGORIES
	CMDOCCATEGORIES
	CMDOCCONTENTS
	CMDOCCONTENTSVERSIONS
	CMDOCFIELDIDS
	CMDOCFIELDS
	CMDOCFIELDVALUES
	CMDOCLAYOUTS
	CMDOCLAYOUTSTYLES
	CMDOCLINKS
	CMDOCTYPES
	CMDOCUMENTS
	CMFOLDERS
	CMLAYOUTDOCUMENTS
	CMREPOSITORIES
Portal	PORTALCATEGORY
Portlet	PORTALPORTLETHANDLES
	PORTALPRODUCERREGISTRY
	PORTALPRODUCERS
	PORTALREGISTRY
	PORTALPORTLETSETTINGS
User	PROFILEGROUPREFERENCES
	PROFILEUSERCONTENTS
	PROFILEUSERFIELDVALUES
	PROFILEUSERMETA
	PROFILEUSERPREFERENCES
	PROFILEUSERS
RSS	RSS

Subsystem	Table
Security	SECURITYACCESSRIGHTS
	SECURITYPERMISSIONMETA
	SECURITYPERMISSIONS
Workflow	WFAUDIT
	WFAUDITLOG
	WFDISPATCH
	WFDOCUMENT
	WFENGINESTATE
	WFFINISHEDPROCESS
	WFMESSAGE
	WFPROCESS
	WFPROCESSSTATE
	WFQUEUE
	WFSUSPENDEDACTIVITIES
	WFWORK
	WFWORKITEM

VI Administering Deployed Applications

Provides information about the Portal Administration Console, runtime logging, the Cache Coordinator, and the Debug subsystem.

21 About the Director Administration Console

This chapter describes how to access the Director Administration Console (DAC) and provides links to information about using specific functions. It includes these sections:

About the DAC

To administer a running exteNd Director application, you use the browser-based Director Administration Console (DAC). The DAC is a predefined exteNd Director Web tier application that you can use immediately after deploying any exteNd Director project that includes the Portal subsystem.

The DAC provides a point-and-click user interface for administering the following subsystems:

- Content Management
- Directory
- Framework
- Portal
- Portlet
- User
- Security
- Workflow

Accessing the DAC

➢ **To access the DAC from the Portal Home page:**

1. Open a Web browser.
2. Go to the Portal Home page by typing an URL that uses this format.

   `http://server/context/portal/`

   For example.

   `http://localhost/ExpressPortal/portal/`
3. Click **Portal Administration** in the upper-right corner of the Portal Home page.
4. Click **Director Administration Console**.

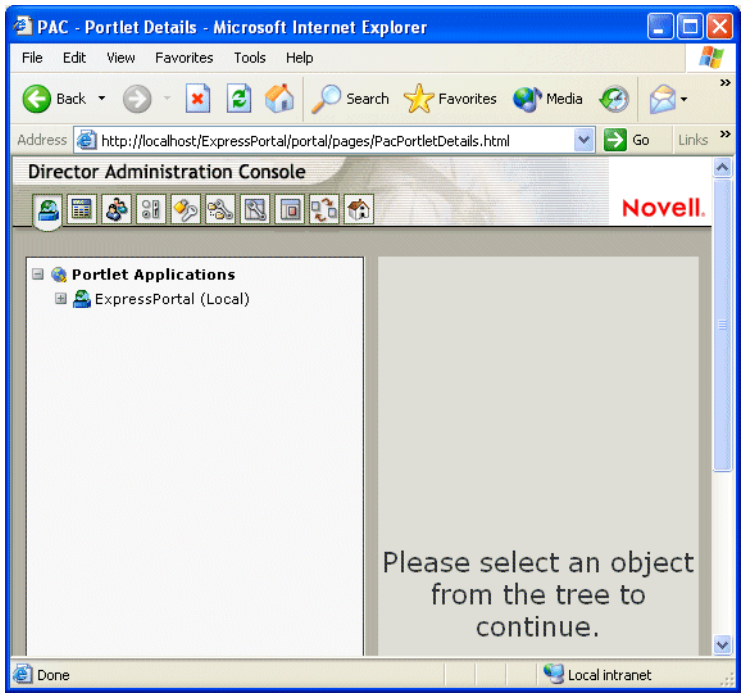
5 If you have not yet logged in, the DAC login page asks for a user name and password:

6 Type an authorized user name and password and click **OK.**

You must log in as a user that has administrative privileges.

NOTE: User names and passwords may be case sensitive depending on how the realm is configured. For information about configuring realms, see the *User Management Guide*.

The DAC displays, with the Portlet Management tab selected:



➢ **To access the Director Administration Console directly:**

1 Open a Web browser.

2 Type an URL that uses this format.

```
http://server/context/PAC/
```

For example.

```
http://localhost/ExpressPortal/PAC/
```

Using the DAC

When the DAC opens in your browser, examine the toolbar:

Each tool administers a specific subsystem (use the links below for information):

Tool	Description	For information see
	Portlet Management	Using the Portlet section of the DAC in the *Portal Guide*
	Portal Management	Using the Portal section of the DAC in the *Portal Guide*
	User Profile Management	Using the Profiles section of the DAC in the *User Management Guide*
	General Configuration	Chapter 22, “Using the General Configuration Section of the DAC” in this book
	Security Management	Using the Security section of the DAC in the *User Management Guide*
	Directory Management	Using the Directory section of the DAC in the *User Management Guide*
	Administration Tools	The help text presented in the Create Director Database Tables portlet
	Content Management	*Content Management Guide*
	Workflow Administration	Workflow Administration in the *Workflow Guide*
	Director Home page	“About Novell exteNd Director” on page 17 in this book

22 Using the General Configuration Section of the DAC

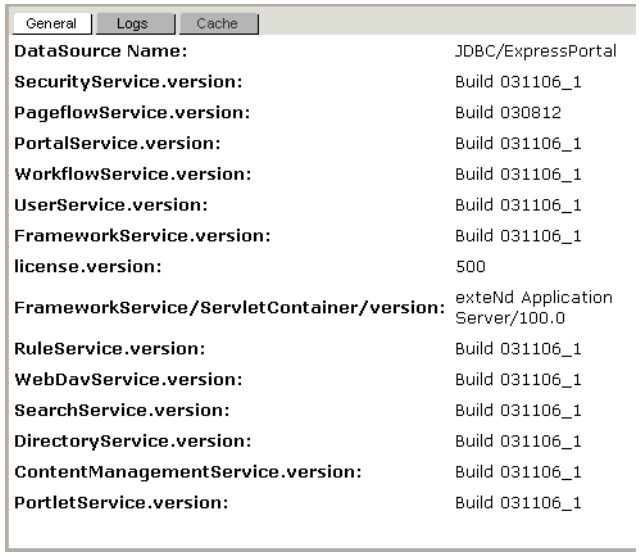
This chapter describes the General Configuration section of the Director Administration Console (DAC), the section that allows you to view and modify settings that control the general behavior of your exteNd Director application. Topics include:

- General
- Logs
- Cache

For information about running the DAC, see Chapter 21, "About the Director Administration Console".

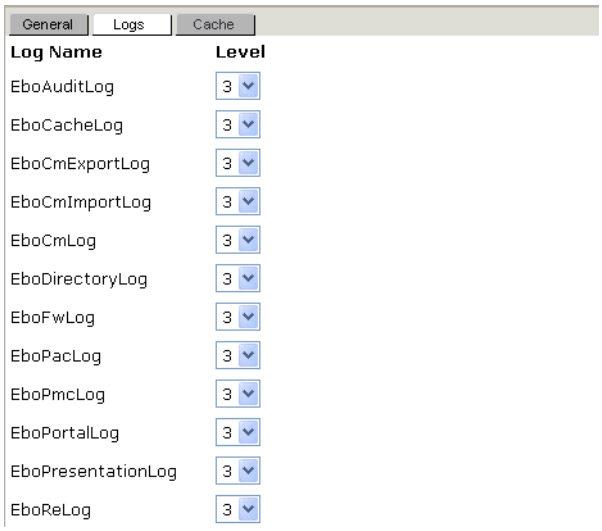
General

The General page displays information about your application. For example, it displays the datasource name for the database in which exteNd Director tables are stored, as well as the build numbers for the subsystems:



Logs

exteNd Director provides a logging facility that writes information in one of several standard logs. After you set up logging, you can use the DAC to set or reset the level of detail for each standard log element:



Various levels of detail determine how much information is written to the log. A value of **0** means no messages are logged, and a value of **5** means all messages are logged.

📖 For more information about setting up logging and detail levels, see Chapter 15, “Logging Information”.

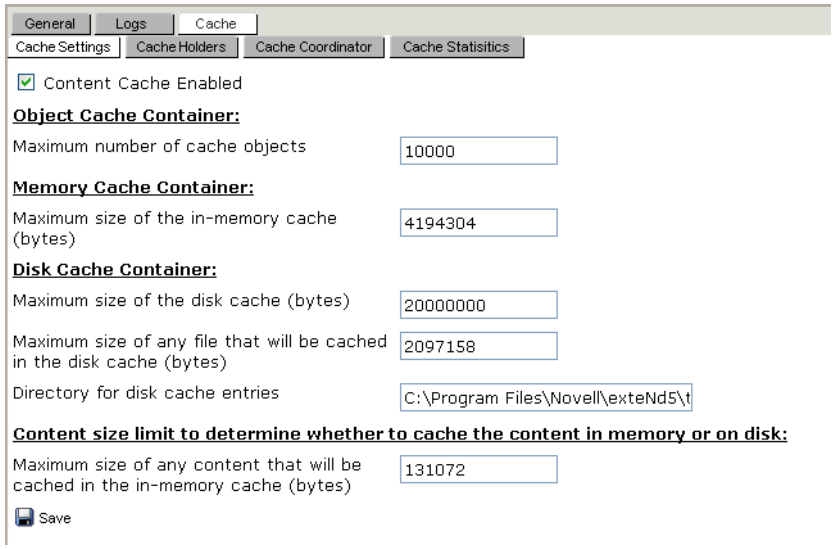
Cache

exteNd Director can use a built-in server-side cache to store reusable, temporary data to optimize performance. An optional Cache Coordinator allows the cache to work within a server cluster. This section describes:

- Cache Settings
- Cache Holders
- Cache Coordinator
- Cache Statistics

Cache Settings

The Cache Settings section of the Configuration panel allows you enable or disable the content cache and set the size of the resources the Cache Manager can control:



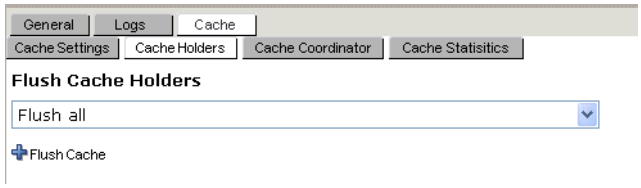
NOTE TO SELF: use file finder thingy here ....

You can use the **Save** button to temporarily change the in-memory cache settings. To change the settings permanently, modify the **config.xml** of the Framework service and redeploy the application.

For more information, see Chapter 14, “Working with Data Caches” and “Cache Statistics” on page 216

Cache Holders

The Cache Holders section of the Configuration panel allows you to flush cached objects from all server-lifetime cache holders, or any specific cache holder:



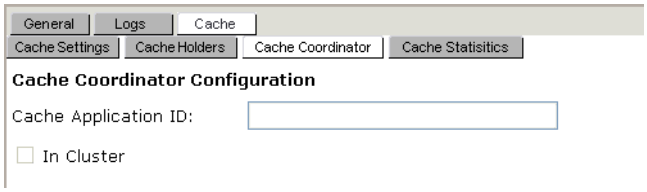
These cache holders are used internally to cache exteNd Director subsystem data for server cluster environments. Click the dropdown list button to see a list of cache holders.

NOTE: These caches are flushed automatically by the system based on how frequently data is used. You can use the Flush option if you have a specific need to flush selected caches or all caches manually.

For more information, see “server-lifetime caching” on page 157.and Chapter 24, “Using the Cache Coordinator”.

Cache Coordinator

The Cache Coordinator section of the Configuration panel displays the Cache Coordinator parameters set at deployment time:

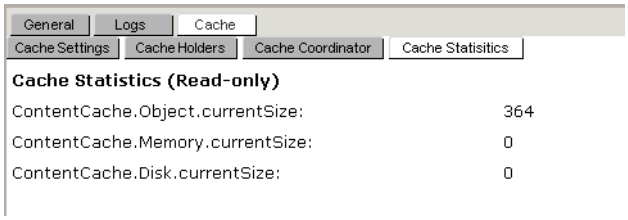


This section applies only if you are running in a server cluster.

For more information, see Chapter 24, "Using the Cache Coordinator".

Cache Statistics

The Cache Statistics section of the Configuration panel displays the current state of the content cache:



There are three cache containers implemented in the Cache Manager:

- Object cache container
- Memory cache container
- Disk cache container

For more information, see Chapter 14, "Working with Data Caches".

Here is a description of the statistics:

Cache statistics item	Description
ContentCache.Object.currentSize:	The number of objects currently stored in the object cache.
ContentCache.Memory.currentSize:	The current total size (bytes) of objects stored in the memory cache. If this item is 0, the memory cache is not being used. it
ContentCache.Disk.currentSize:	The current total size (bytes) of objects stored in the disk cache. If this item is 0, the disk cache is not being used. it

If your portal application uses only the object cache container, you can check ContentCache.Object.currentSize and, if necessary, increase the maximum object size in the cache configuration to allow better performance. That value is defined in the **config.xml** of the Framework service:

```
ContentCache.Object.maxSize
```

NOTE: To make a persistent change to this value, you must redeploy the application. A change you make from the DAC lasts only until the application server restarts.

23 Using the Debug Subsystem

This chapter explains how to use the **Debug** subsystem of exteNd Director. Topics include:

About the Debug subsystem

The Debug subsystem helps you troubleshoot a deployed exteNd Director application by providing reports about the application and its server environment. These reports include:

Report	Description
exteNd Director Resources	Queries the subsystem configuration and service entries for your application to display their current (in-memory) values Also lets you validate resource set bindings.
HTTP Resources	Displays information about the application's deployed servlets and JSP pages
JNDI Resources	Displays the tree of JNDI resources available to your application on its J2EE server
exteNd Director Archive Resources	Lists the archives deployed to an exteNd Application Server and lets you display archive details (contents, unresolved JAR references)

How it works

The Debug subsystem consists of a set of JSP pages that display various debugging reports, along with a supporting tag library (**debug.tld**) and underlying classes:

- **index.jsp** is the Debug subsystem's home page.
- **debugEbo.jsp** displays the exteNd Director Resources report. It uses the custom tag DebugEbo defined in debug.tld.
- **debugResources.jsp** displays the HTTP Resources report. It uses the custom tag DebugResources defined in debug.tld.
- **debugJNDI.jsp** displays the JNDI Resources report. It uses the custom tag DebugJNDI defined in debug.tld.
- **debugArchives.jsp** displays the Novell Archive Resources report. It uses the custom tag DebugArchive defined in debug.tld.

In an EAR project, the Debug subsystem is packaged in the **Debug.war** file. In a WAR project, the resources required for the subsystem are located in appropriate places within the WAR. For example, the JSP pages are located in the **pages** folder at the top level of the WAR.

Security considerations

The Debug subsystem exposes a lot of information about the environment it runs in. By default, this subsystem is not secured.

Typically, you should not deploy the Debug subsystem to a production environment. If you choose to do so, you should at least restrict the **/Debug/** URI (by editing the Debug web.xml file) so that only authorized users can access it.

Setting up the Debug subsystem

➢ **To use the Debug subsystem in your exteNd Director application:**

1 Add the Debug subsystem to your exteNd Director project.

When you create a new exteNd Director project via **File>New Project** and request a typical setup, the Debug subsystem is automatically included. If you request a custom setup, you'll need to select the Debug subsystem explicitly.

You can later modify your exteNd Director project via **Project>exteNd Director Project>Setup** to add or remove the Debug subsystem.

2 Build, archive, and deploy the project to your J2EE server.

You should now be able to run the Debug subsystem.

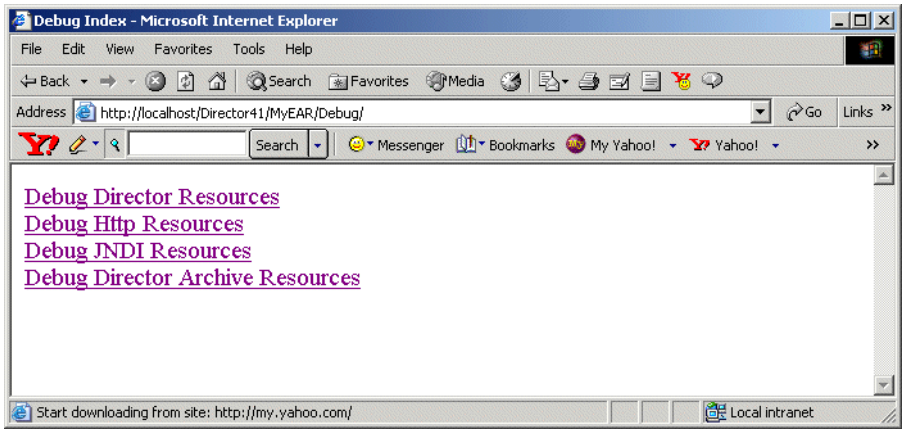
Running the Debug subsystem

Once you've deployed the Debug subsystem as part of your exteNd Director application, running it is just a matter of using your Web browser to visit one or more of the Debug pages:

- Going to the Debug home page
- Reporting on exteNd Director resources
- Reporting on HTTP resources
- Reporting on JNDI resources
- Reporting on exteNd Director archive resources

Going to the Debug home page

The Debug subsystem provides a home page that displays links to the Debug reports:



➢ **To go to the home page of the Debug subsystem:**

1 In your Web browser, type one of the following URLs:

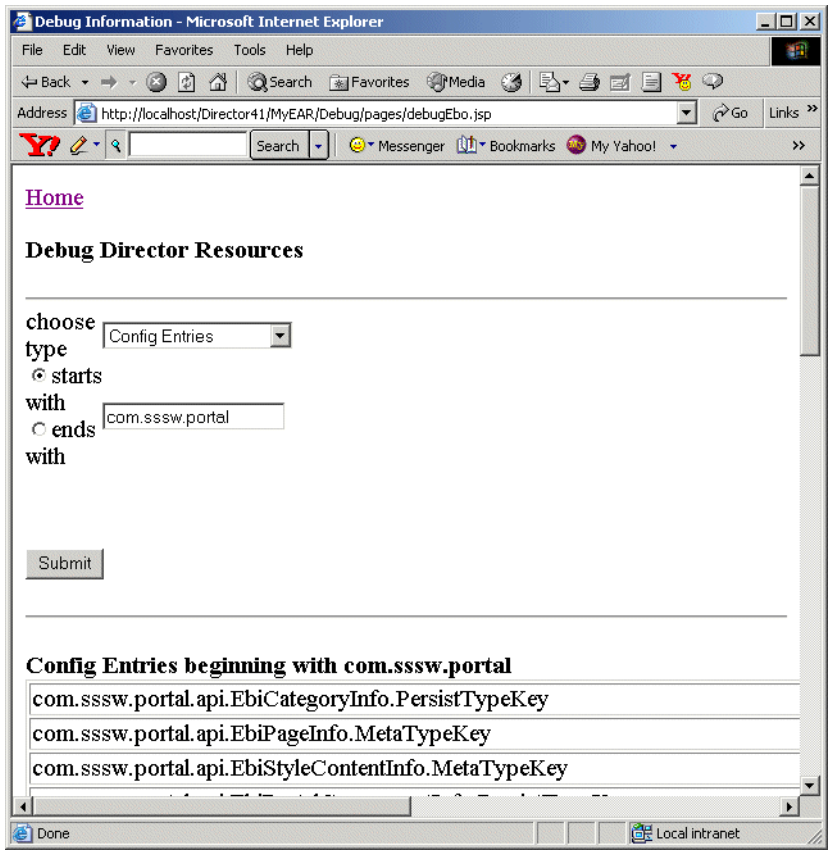
If you've deployed to	The URL is
An exteNd Application Server	http://*server*/*database*/*ear-namespace*/Debug/ For example: `http://localhost/MyDirectorDB/MyDirectorApp/Debug/`
Another J2EE server	http://*server*/*ear-namespace*/Debug/ For example: `http://localhost/MyDirectorApp/Debug/`

2 When the Debug home page appears, click the links to view Debug reports.

Reporting on exteNd Director resources

You can use the exteNd Director Resources report to do any of the following:

- **Display subsystem configuration entries** for your application.
  By default the report lists all entries, but you can optionally select just those that start or end with a specified string. The current (in-memory) value of each entry is shown. (These values are loaded from the subsystem **config.xml** files, but may then be modified temporarily via API calls.)
- **Display subsystem service entries** for your application.
  By default the report lists all entries, but you can optionally select just those that start with a specified string. The current (in-memory) value of each entry is shown. (These values are loaded from the subsystem **services.xml** files, but may then be modified temporarily via API calls.)
- **Validate ResourceSet bindings** for your application.



The Debug subsystem also provides an alternative way to get this report information—the **echo servlet**.

➢ **To report on your application's exteNd Director resources:**

1 From the Debug home page, click **Debug Director Resources**.

The exteNd Director Resources report page displays.

2 Use the **choose type** dropdown to select which kind of information you want the report to display:

- Config Entries
- Service Entries
- ResourceSet Entries

3 If you select Config Entries or Service Entries, you can optionally narrow the report query. To display only those entry names that contain a particular string:

- Fill in the **text box** with that string (for example, `com.sssw.portal`).
- Select the **starts with** or **ends with** radio button to specify where to look for that string in entry names.

NOTE: For Service Entries, you can only select **starts with**.

4 Click the **Submit** button.

The report results display on the page.

➢ **To use the echo servlet:**

1 In your Web browser, start typing the echo servlet URL as follows:

If you've deployed to	Start the URL with
An exteNd Application Server	http://*server*/*database*/*ear-namespace*/Debug/echo? For example: http://localhost/MyDirectorDB/MyDirectorApp/Debug/echo?
Another J2EE server	http://*server*/*ear-namespace*/Debug/echo? For example: `http://localhost/MyDirectorApp/Debug/echo?`

2 Finish typing the URL by adding information about the report to generate:

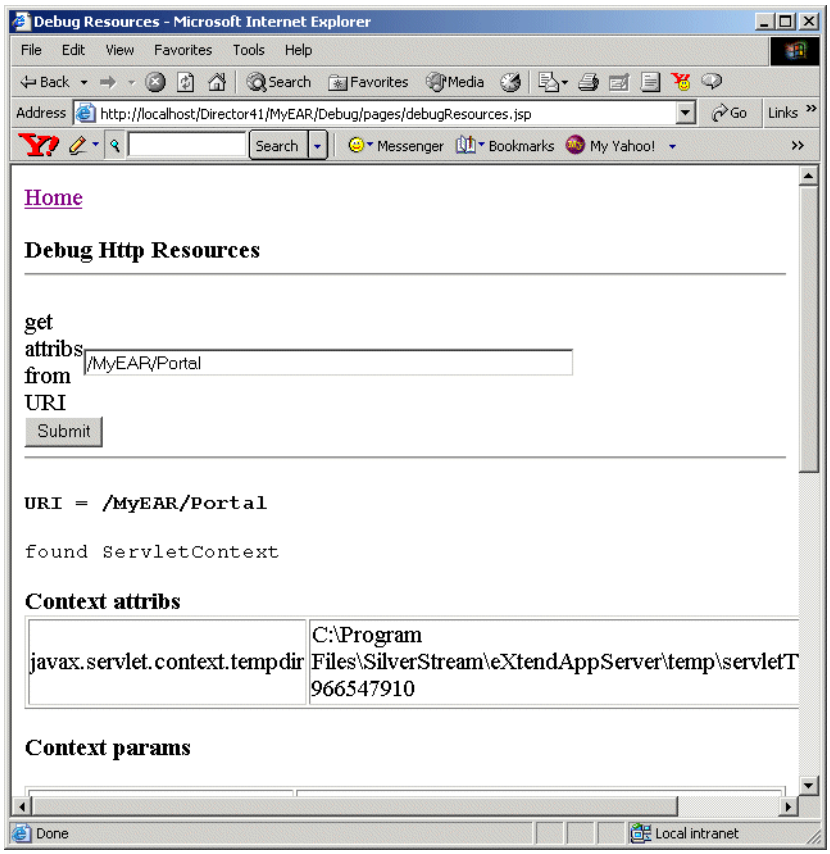
If you want to	Add this to the URL
Display subsystem configuration entries	One of the following: ◆ To get configuration entry names that **start with** a particular string: `type=c&sw=`*`string`* ◆ To get configuration entry names that **end with** a particular string: `type=c&ew=`*`string`* ◆ To get **all** configuration entry names: `type=c&sw=`
Display subsystem service entries	One of the following: ◆ To get service entry names that **start with** a particular string: `type=s&sw=`*`string`* ◆ To get **all** service entry names: `type=s&sw=`
Validate resource set bindings	`type=rs`

For example, this URL invokes the echo servlet to display subsystem configuration entries that start with com.sssw.portal:

`http://localhost/MyDirectorDB/MyDirectorApp/Debug/echo?type=c&sw=com.sssw.portal`

Reporting on HTTP resources

You can use the HTTP Resources report to display information about the servlets and JSP pages mapped to particular URIs in your application. The report gets this information by using the Servlet API:



➢ **To report on your application's HTTP resources:**

1 From the Debug home page, click **Debug Http Resources**.

The HTTP Resources report page displays.

2 Fill in the **get attribs from URI** text box with the mapping you want information about.

Start with the EAR namespace you've established for your application. In this example, the EAR namespace is MyDirectorApp:

```
/MyDirectorApp/Portal/
```

3 Click the **Submit** button.

The report results display on the page.

Reporting on JNDI resources

You can use the JNDI Resources report to display a tree of the JNDI resources available to your application on its J2EE server. Code in your exteNd Director application can access these resources via a JNDI lookup.

This report works best with an exteNd Application Server.

➢ **To report on your J2EE server's JNDI resources:**

- From the Debug home page, click **Debug JNDI Resources**.

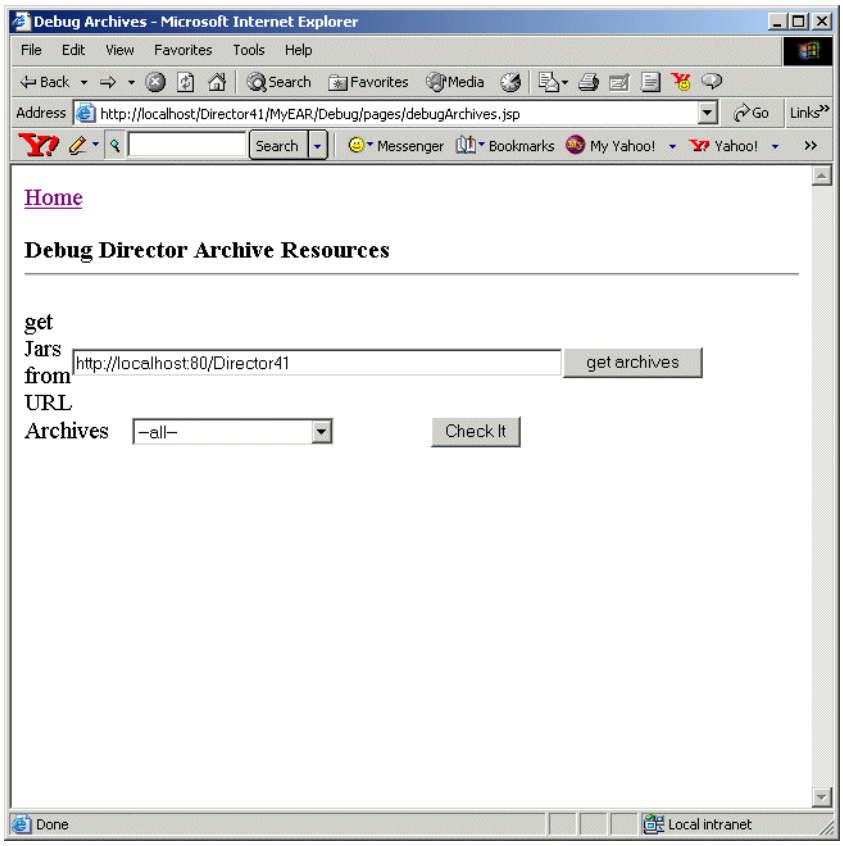
  The JNDI Resources report page displays, including its results.

Reporting on exteNd Director archive resources

You can use the exteNd Director Archive Resources report to list the archives deployed to an application server. You can then use that list to select archives you want to display details about. Archive details provided by this report include:

- **Contents** (classes, other archives, and so on)
- **Unresolved JAR references**

  The report checks for JAR files that are referenced in the manifest classpath but are not available in the classpath on the server. Such unresolved references can cause a ClassNotFoundException at runtime. The report displays them in red to warn you.



➢ **To report on your exteNd Director archive resources:**

1. From the Debug home page, click **Debug** **exteNd Director** **Archive Resources**.

   The exteNd Director Archive Resources report page displays.
2. Fill in the **get Jars from URL** text box to point to the appropriate server and database. For example:

   ```
   http://localhost:80/MyDirectorDB
   ```
3. Click the **get archives** button.

   This populates the Archives dropdown with the current list of archives from that location.
4. Use the **Archives** dropdown to select an item you want details about.

   You can select a particular archive or **-all-** (for details on every archive).
5. Click the **Check It** button.

   The archive details display on the page.

24 Using the Cache Coordinator

This chapter describes how to set up the exteNd Director Cache Coordinator for server cluster configurations. It has these sections:

📖 For background information, see Chapter 14, “Working with Data Caches”.

About the Cache Coordinator

exteNd Director provides a **Cache Coordinator** for managing cached data in server cluster configurations. Specifically, the Cache Coordinator manages objects stored in the *server-lifetime cache*. (see Chapter 14, “Working with Data Caches”). The Cache Coordinator runs on one machine in the server cluster.

How the Cache Coordinator works

You may store any objects in a server-lifetime cache holder. However, only the objects that are instantiated from data from a persistent store (such as a database) can be synchronized in the cluster environment.

All servers in a server cluster access the same persistence store. Any cached objects that are instantiated from the persistence store can be synchronized among application servers that are running in a clustered environment. Here’s how it works:

- When data (or an object) has been updated in the database, the cached object is updated accordingly in the affected server’s lifetime cache holder. At this point the server’s Cache Manager triggers a cache invalidation event that sends a notification to the Cache Coordinator.
- The Cache Coordinator broadcasts a message to the rest of the registered server instances to tell them that this data has been changed.
- After receiving the message, each server flushes its server-lifetime cache.
- The next time this data is requested, each server instantiates the updated object by fetching it from the persistent store, and recaches it in its server-lifetime cache. This ensures that the cached objects are always up-to-date.
- The Cache Coordinator also periodically verifies that each cluster server is running, removing from the cluster any server that has failed.

NOTE: Data is not replicated to different server instances. This means if you store temporary data in the server lifetime cache, the data will not be replicated to other server instances.

Triggering a cache invalidation event

If you want to cache an object that will automatically trigger an invalidation event, use EbiSrvLifetimeCacheHolder.putObjectInCache() and pass in EboState.UPDATE. Calling EbiSrvLIfetimeCacheHolder.removeObjectInCache() always triggers the cache invalidation event.

IMPORTANT: Do not remove objects from cache if those objects are not physically removed from persistent storage.

If you want to cache an object without triggering the cache invalidation event, use EbiSrvLifetimeCacheHolder.putObjectInCache() and pass in EboState.SILIENCE.

 For more information, see EbiSrvLifetimeCacheHolder in the *API Reference*.

NOTE: exteNd Director provides built-in server-lifetime cache holders for many types of subsystem runtime data, as described in "Built-in cache holders" on page 157.

Reconfiguring the Cache Coordinator

By default, the Novell exteNd Installation Wizard installs the Cache Coordinator. (You can override this behavior using a custom installation). Here is a list of Cache Coordinator properties that you can reconfigure and where you can access each property.

Setting	Description	Editable in
Server Host	The name of the host machine the Cache Manager runs on. The default value is **localhost**. **IMPORTANT:** You must change the hostname from **localhost** to the actual host name for the Cache Coordinator server. The **localhost** entry will disable the cache invalidation feature.	FrameworkService config.xml
Server Port	RMI registration port the Cache Coordinator listens on. The default value is **54490**. You must specify the same value when you create an exteNd Director project.	SilverCache.props FrameworkService config.xml
Application Identifier	UID for the exteNd Director application. This is created automatically in the exteNd Director Project Wizard when you select clustering. **IMPORTANT:** If you plan to use the shared library feature in a server cluster, each server must point to a single ConfigService.jar that includes the cluster properties.	Not editable
Start retry count	Number of times the server application instance attempts to connect with the Cache Coordinator before generating an error.	FrameworkService config.xml
Start retry interval	Number of seconds before each connection retry.	FrameworkService config.xml

Setting	Description	Editable in
Watcher sleep interval	Number of seconds the Cache Coordinator waits between attempts to contact the server application instance. If an application fails, the Cache Coordinator drops it from the cluster.	SilverCache.props
Reconnect sleep interval	How long after a connection failure the Cache Coordinator will wait before attempting to reconnect.	FrameworkService config.xml

➢ **To edit Cache Coordinator properties in config.xml:**

1 In your exteNd Director project, open **config.xml** in the FrameworkService-conf directory.

2 This opens the XML Editor. Find the property you want and edit the value.

NOTE: The following shows the source view of the properties. You can also use the Graphical (default) view in the editor. Choose **Graphical View** or **XML Source View** in the editor.

```
</property>
    <property>
        <key>com.sssw.fw.coordinator.Host</key>
        <value>myCMServer</value>
    </property>
    <property>
        <key>com.sssw.fw.coordinator.Port</key>
        <value>54490</value>
    </property>
    <property>
        <key>com.sssw.fw.coordinator.start.sleepInterval</key>
        <value>10</value>
    </property>
    <property>
        <key>com.sssw.fw.coordinator.Start.tryCount</key>
        <value>3</value>
    </property>
    <property>
        <key>com.sssw.fw.coordinator.reconnect.sleepInterval</key>
        <value>60</value>
    </property>
```

3 Save the file and deploy the project.

➢ **To edit properties in SilverCache.props:**

1 Open **SilverCache.props**, located at:

installdir/exteNdDirector/bin/SilverCache.props

2 Find the property you want and edit the value:

```
# This properties file contains properties for the
# extendDirector Cache Coordinator
# version 4.0 or later

cc.port=54490
cc.watcher.interval=60
cc.logging.level=3
```

3 Save the file and run the Cache Coordinator, as described next.

Running the Cache Coordinator

➢ **To run the Cache Coordinator:**

- Depending on your platform, do one of the following:

Platform	What to do
Windows	**1** Make sure the server cluster is running. **2** Go to **Start>All Programs>Novell exteNd *n.n*>Director>Director Cache Coordinator**.
Windows, UNIX, or Linux	**1** Go to the Novell exteNd installation directory. **2** Run the executable file named **SilverCacheCoordinator** in the **Director\bin** directory.

➢ **To run the Cache Coordinator as a service (Windows only):**

1 Go to the **Services** panel.

In Windows NT choose **Services** from **Control Panel>Administrative Services**.

In Windows 2000 choose **Services** from **Control Panel>Administrative Tools**.

2 Select **Silver cache coordinator**.

3 Select **Start** to manually start the program, or select **Startup** and change the service to **automatic** so it will automatically run in the background when you start the server cluster.

Recovering from a Cache Coordinator failure

If a server in the cluster cannot connect to the Cache Coordinator, the server disables the cache invalidation function and reconnects to the Cache Coordinator. If the reconnect fails, you can flush the cluster server caches without restarting the servers. You can access each server in the cluster from the Director Administration Console (DAC) and use the **Flush All** option to:

- Clean up the caches
- Reconnect to the Cache Coordinator instance in the cluster

NOTE: This option does not apply to a server cluster running on the WebSphere application server. You must restart each server in the cluster.

If you restart the servers in the cluster, then flushing the cache is not necessary: the servers start with nothing cached.

 For more information about accessing cache information using the DAC, see Chapter 22, “Using the General Configuration Section of the DAC”.

Logging Cache Coordinator activity

exteNd Director provides logging facilities for Cache Coordinator activity. You can monitor cache invalidation messages sent from each server instance to the Cache Coordinator, and monitor activity on the Cache Coordinator itself.

To set up cache invalidation logging you need to do the following:

- Update the logging level to **5** for each server instance and for the Cache Coordinator.
- To make logging more legible, provide a server identifier for each server instance.

NOTE: A ServerID is provided automatically with some application servers. See "Providing server identifiers" on page 229.

📖 For general information about logging and logging levels, see Chapter 15, "Logging Information".

Updating the logging level for server instances and the Cache Coordinator

Cache invalidation logging uses Cache logging level **5**.

You can set the log level two ways:

To change the logging level permanently Change the **EboCacheLog.LoggingLevel** property (in FrameworkService-conf /config.xml) to **5**. Updating this property value will persist the change for subsequent server sessions.

📖 For information on editing this file, see "To edit Cache Coordinator properties in config.xml:" on page 227.

To change the logging level for the server session Change the logging level using the DAC. Making this change allows you to monitor cache invalidation messages sent from this server instance to the Cache Coordinator for the current server session.

📖 For more information, see Chapter 22, "Using the General Configuration Section of the DAC".

Updating the logging level for the Cache Coordinator

You also need to update the logging level to 5 on the Cache Coordinator server. Use one of these methods:

To change the logging level permanently Change the **logging.level** property in SilverCache.props to **5**. Updating this property value will persist the change for subsequent server sessions.

📖 For information on editing this file, see "To edit properties in SilverCache.props:" on page 227.

To change the logging level for the server session Run the Cache Coordinator in debug mode by adding an option switch:

```
SilverCacheCoordinator -l 5
```

This will provide information about how many application instances are registered with the Cache Coordinator.

TIP: For more information about command options, run:

```
SilverCacheCoordinator -?
```

Providing server identifiers

In order to allow better logging and have a readable server ID for each registered server instance, certain application servers provide an attribute that clearly identifies each server instance. As an alternative you can use the -D server startup option

Support for exteNd Application Server There is an attribute available from ServletContext called **com.novell.appsrv.servlet.application.host:port** that automatically provides a different Server ID for each server instance.

Support for WebSphere Application Servers There is an attribute available from ServletContext called **com.ibm.websphere.servlet.application.** that automatically provides a different ServerID (**host:***port#*) for each server instance.

Server startup option You can use the **+D** server startup option to provide your own n identifier for each server instance. Specify a value for the **ServerID** attribute, as shown in this example:

```
Silverserver +Dcom.novell.afw.ServerID=myID
```

where myID represents a unique identifier.

NOTE: This option overrides any identifiers specified elsewhere.

The ServerID attribute is combined with a Cache Manager ID that is an automatically generated UUID for each server restart.

NOTE: If a ServerID is not specified only the Cache Manager ID is used. Since this value is difficult to read in the context of logging, it is not recommended.

About the logging messages

Logging messages are generated on the sever instances and the Cache Coordinator whenever a server is started and when server instances generate cache invalidation messages.

Server startup logging

A message is generated on each serve instance when it attempts to register with the Cache Coordinator at startup. Here is what the logging message looks like on a server instance:

```
EboCacheLog|5|11/11/03|15:20:09:113|<Connect> to the cache coordinator @ michigan:54490...
EboCacheLog|5|11/11/03|15:20:09:154|<Cache Coordinator Service rmi URI> is: 'rmi
 ://michigan:54490/EboCacheCoordinator/exteNdDirectorCacheCoordinator'.
EboCacheLog|5|11/11/03|15:20:09:424|<Register> this exteNd Director instance with the
  cache coordinator...
EboCacheLog|5|11/11/03|15:20:09:464|<Connect OK> connected to the cache coordinator
   successfully.
```

A corresponding message is generated on the Cache Coordinator. The serverID in the example is **clone1.** The client count (which is **1** in the example) indicates the number of servers currently connected:

```
EboCacheLog|5|11/11/03|15:20:09:464|<Register> a client [ServerID: <clone1:
  c373e9f8cc736a65b1c7444553544200> AppID: <c373e9f8cc46d5b9b9d4444553544200>].

EboCacheLog|4|11/11/03|15:20:09:464|<Registered client count> for [AppID:
  <c373e9f8cc46d5b9b9d4444553544200>] is: 1.
```

Cache invalidation message logging

The first example shows output triggered on the server instance when putObjectInCache() is called (with state=EboState.UPDATE) from an EbiSrvLifetimeCacheHolder:

```
EboCacheLog|5|11/11/03|15:27:01:416|<Send notification> from [ServerID: <clone1:
c373e9f8cc736a65b1c7444553544200> AppID: <c373e9f8cc46d5b9b9d4444553544200>]
to cache coordinator, message: cached content with [key: <admin> cache holder ID: <
DirectoryService.UserCacheHolder.exteNd Server>] has been modified [state 113].
```

Here is the corresponding output on the Cache Coordinator server console:

```
EboCacheLog|5|11/11/03|15:27:01:426|<Receive notification> from
[ServerID: <clone1:c373e9f8cc736a65b1c7444553544200> AppID: <c373e9f8cc46d5b9b9d4444553544200>]
message: cached content with [key: <admin> from cache holder: <DirectoryService.
UserCacheHolder.exteNd Server>] has been modified [state: 113].
```

The second example shows output triggered when the removeObjectInCache() method is call from EbiSrvLifetimeCacheHolder:

```
EboCacheLog|5|11/11/03|15:32:11:572|<Send notification> from [ServerID: <clone1:
```

```
c373e9f8cc736a65b1c7444553544200> AppID: <c373e9f8cc46d5b9b9d4444553544200>] to
cache coordinator, message: cached content with [key: <PacPortletDetails.html> c
ache holder ID: <testCluster.PageCacheHolder>] has been removed [state 112].
```

Here is the corresponding output from the Cache Coordinator server console:

```
EboCacheLog|5|11/11/03|15:32:11:592|<Receive notification> from
[ServerID: <clone1:c373e9f8cc736a65b1c7444553544200> AppID: <c373e9f8cc46d5b9b9d4444553544200>]
message: cached content with [key: <PacPortletDetails.html> from cache holder: <
testCluster.PageCacheHolder>] has been removed [state: 112].
```

Remote Cache Coordinator administration

You can get an EbiRemoteCoordinatorAdmin object for the Cache Manager by calling EbiCacheManager.getRemoteCoordinatorAdmin(). This object allows you to administrator the Cache Coordinator from any application server instance. For example, you can get the list of registered server instances that are currently connecting to the Cache Coordinator. You might implement a servlet or a portlet to perform remote administration

 For more information, see EbiRemoteCoordinatorAdmin in *API Reference*.

VII Third-Party Tools

Explains how to use Macromedia Dreamweaver with exteNd Director applications

25 Using Dreamweaver with exteNd Director

This chapter describes how to use Macromedia Dreamweaver with exteNd Director. It includes these sections:

- About exteNd Director and Dreamweaver
- Installing Dreamweaver extensions
- Using the exteNd Director Integration extension

This chapter assumes that you are familiar with Dreamweaver.

About exteNd Director and Dreamweaver

Dreamweaver is an integrated development environment for creating Web pages and creating and managing Web sites and Internet applications . exteNd Director ships with the Novell exteNd Integration extension that lets you use Dreamweaver with exteNd Director applications to:

Use Dreamweaver to	Description
Add portlets and components to PID (Portal ID) pages	The Novell exteNd Director Integration (Novell_Director.mxp) extension PID pages are the only type of exteNd Director pages that can be used with this extension 📖 For instructions, see "Using the exteNd Director Integration extension" on page 236
Edit files in the Content Management repository	Requires you to configure: • WebDAV access from Dreamweaver's Site Manager to the Content Management repository. 📖 For instructions on using WebDAV with an exteNd Director application, see Dreamweaver's Site Manager documentation and the WebDAV chapter in the *Content Management Guide* • The exteNd Director application so that PID pages are stored in the Content Management repository, not their typical resource set location 📖 For instructions, see "Displaying PID pages from Content Management" on page 238

NOTE: This extension works with both Dreamweaver and Dreamweaver UltraDev.

Installing Dreamweaver extensions

You'll use Macromedia's Extension Manager to install the exteNd Director Dreamweaver extension.

➢ **To install the exteNd Director Dreamweaver extension:**

1 Open the Macromedia Extension Manager.

2 Choose **File>Install Extension**.

3 Navigate to the directory containing the extension you want to install:

Extension	File location
Novell exteNd Director Integration	Director\Utilities\Dreamweaver\Novell_Director.mxp

4 Choose the .mxp file associated with the extension you want to install.

5 Click **Install**.

6 Follow the extension manager's installation instructions.

Using the exteNd Director Integration extension

The exteNd Director Integration extension allows the Dreamweaver work area to:

- Insert portlets or components into exteNd Director PID pages by selecting from a list obtained from a deployed exteNd Director application. See Inserting component tags next.
- Use WebDAV to access the Content Management repository of a deployed exteNd Director application.

Using these two features together requires configuring the exteNd Director application before deployment. See "Displaying PID pages from Content Management" on page 238.

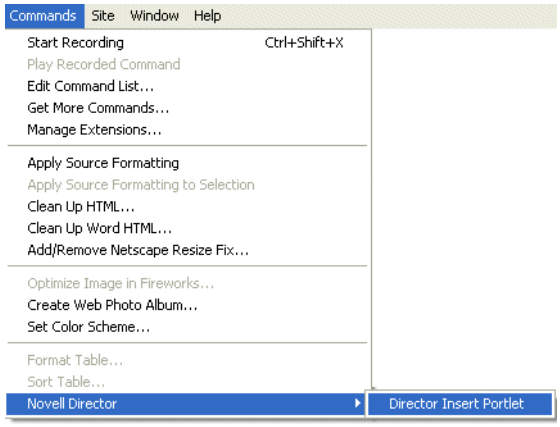
Inserting component tags

The exteNd Director Integration extension allows the Dreamweaver work area to connect to a deployed exteNd Director portal application and insert s3-component tags into PID pages. You can select from a list of portlets or components in the Portal WAR of the deployed application.

➢ **To insert a component tag:**

1 In Dreamweaver **Design View**, put the cursor where you want to insert the portlet or component.

2 Select **Commands>Novell Director>Director Insert Portlet**.



3 In the **Director Insert Portlet**, specify:

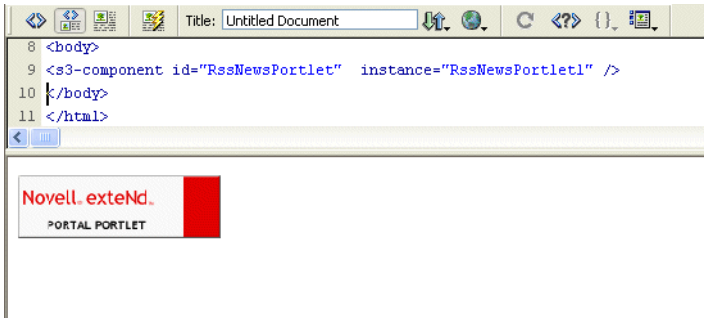
Setting	What to specify
Hostname/IP:port	The host name (or IP address) and optionally the port number of the application server your exteNd Director application has been deployed on. For example, localhost.
Database	The name of the exteNd Director application database on the application server. For example, SilverMaster50. (Leave this blank for servers that do not use a database.)
Portal Location (Portal or namespace/Portal)	The exteNd Director application's namespace (if the application is configured with one) followed by the name of the portal. For example, ExpressPortal.

4 Click the **Get Portlet List**.

The dialog displays a dropdown list box that shows all of the available portlets and components in the target portal.

5 Select a portlet or component and click **Insert Portlet**.

The extension inserts an s3-component tag into the PID page HTML code and displays a corresponding icon at the specified location in the design view:



Displaying PID pages from Content Management

By default, PID pages are not stored in the Content Management repository of an exteNd Director application; they are stored in a resource set. For that reason you must add a new **EboResourceServlet** that can fetch and display PID pages from the Content Management repository.

➢ **To configure the exteNd Director application:**

1 **Before** deploying your exteNd Director project, do the following:

1a Open **web.xml** for the Portal WAR.

1b Add the following lines of XML code to the file:

```
<servlet>
    <servlet-name>cmresources</servlet-name>
    <servlet-class> com.sssw.cm.servlet.EboResourceServlet</servlet-class>
    <init-param>
      <param-name>enable-pid-support</param-name>
      <param-value>true</param-value>
    </init-param>
  <init-param>
      <!--This is the directory in the cm system that -->
      <!--stores pids-->
      <param-name>pid-path</param-name>
      <param-value>/cmpages</param-value>
    </init-param>
</servlet>
    <servlet-mapping>
    <servlet-name>cmresources</servlet-name>
    <url-pattern>/cmresources/*</url-pattern>
    </servlet-mapping>
```

Insert the tags immediately following the other `<servlet>` and `<servlet-mapping>` tags.

📖 See "EboResourceServlet initialization parameters" on page 239 for an explanation of the code.

1c Build and deploy (or redeploy) the exteNd Director application.

2 **After** deploying your exteNd Director project, do the following:

2a Set your Dreamweaver WebDAV connection's remote site folder to `/cmpages`.

2b Use the CMS Administration Console to move the PID pages of interest from the resource set of your exteNd Director project to the Content Management repository of the deployed application (or create new ones).

2c Use the CMS Administration Console to move the images and other objects used by each PID page; this is necessary because the resource servlet does not modify the relative URLs contained in a page.

📖 For information about using the CMS Administration Console, see the *Content Management Guide*.

➢ **To access PID pages in the Content Management repository:**

- Specify the `servlet-name` and the `pid-path` folder in the URL. For example:

  `http://localhost/Director/Portal/cmresources/cmpages/myPID.html`

NOTE: To access a PID page in the Content Management repository from another PID page, use the CMPIDReader component.

CMPIDReader accepts a single parameter CMPATH that represents the path to the PID page that is being retrieved from the Content Management repository. The CMPATH can be sent via an s3-component tag or an HTML request. Examples:

```
<s3-component ID="CMPIDReader" NAME="CMPIDReader" CMPATH="/cmpages/myPID.html" />

http://localhost/Director/Portal/main/comp/CMPIDReader?cmpath=%2fcmpages%2fmyPID.html
```

EboResourceServlet initialization parameters

The Content Management resource servlet (**EboResourceServlet**) provides URL access to any object in the Content Management repository via path or document ID. By adding the initialization parameters shown below, you can enable this servlet to also render PID pages stored in the Content Management subsystem.

The Content Management resource servlet has two initialization parameters:

Initialization parameter	Description
`enable-pid-support`	If enable-pid-support is true, the resource servlet checks the incoming path of the Content Management document. If it matches the path set under pid-path, the resource servlet assumes that the HTML document is a PID page and processes it using the MIME type designated by the Content Management subsystem. Objects in all other path locations are returned as is.
`pid-path`	The pid-path designates one specific folder in the Content Management repository as storage for PID pages. Restricting PID pages to a single folder is necessary for performance reasons. Content Management folders potentially contain vast numbers of large documents, and scanning every document for s3-component tags would be quite inefficient.

VIII Reference

Provides general reference information for developers using exteNd Director

- Chapter 26, “Project File Locations”

26 Project File Locations

A typical exteNd Director project contains a large set of editable subsystem configuration files and application resources. This chapter helps you quickly locate different types of files in your EAR or WAR project.

NOTE: For convenient access, a link to this document is available in the contents pane of the help system.

This chapter has these sections:

Related documentation

Understanding project disk structure This chapter assumes you have a basic familiarity with the exteNd Director product architecture and design patterns.

Understanding project resources This chapter assumes you have a basic familiarity with the exteNd Director project resources.

For information, see Chapter 6, “Using the Resource Set in an exteNd Director Application”.

Using resource management to locate files exteNd Director provides a set of tools that can be very useful for locating project files. For information, see:

- Chapter 8, “Using the Relationship Viewer”
- Chapter 9, “Searching a Resource Set”
- Chapter 10, “Working with Views”

  TIP: Predefined *views* can be very useful for locating project files within a resource set. You can construct your own views by saving search results and/or editing view definition files.
- “Using predefined views” on page 101

Configuration files

Content Management subsystem configuration files

Project archive	Layout type	Location
EAR	Archive	*project_name*.spf\library\ConfigService.jar\ ContentMgmtService-conf**config.xml**
	Source	*project_name*.spf\library\ConfigService\ConfigService.spf\ContentMgmtService-conf**config.xml**
WAR	Archive	*project_name*.spf\WEB_INF\lib\ ConfigService.jar/ContentMgmtService-conf**config.xml**
	Source	*project_name*.spf\WEB_INF\lib\ConfigService\ConfigService.spf\ ContentMgmtService-conf**config.xml**

CM task list configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name*.spf\library\ConfigService.jar\ ContentMgmtService-conf**Default_tasklist.xml**
	Source	*project_name*.spf\library\ConfigService\ConfigService.spf\ContentMgmtService-conf**Default_tasklist.xml**
WAR	Archive	*project_name*.spf\WEB_INF\lib\ ConfigService.jar\ContentMgmtService-conf**Default_tasklist.xml**
	Source	*project_name*.spf\WEB_INF\lib\ConfigService\ConfigService.spf\ ContentMgmtService-conf**Default_tasklist.xml**

CM task types configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name*.spf\library\ConfigService.jar\ ContentMgmtService-conf**tasktypes.xml**
	Source	*project_name*.spf\library\ConfigService\ ConfigService.spf\ContentMgmtService-conf**tasktypes.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ ConfigService.jar\ContentMgmtService-conf**tasktypes.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\ ContentMgmtService-conf**tasktypes.xml**

Directory subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\DirectoryService-conf**config.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\DirectoryService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\DirectoryService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\DirectoryService-conf**config.xml**

Framework subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\FrameworkService-conf**config.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\FrameworkService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\FrameworkService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\libConfigService\ConfigService.spf\FrameworkService-conf**config.xml**

Novell-portlet configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_App_name*.war\WEB-INF**novell-portlet.xml**
	Source	*project_name.spf**Portal_app_name*.spf\WEB-INF**novell-portlet.xml**
WAR	Archive	*project_name.spf*\WEB-INF**novell-portlet.xml**
	Source	*project_name.spf*\WEB-INF**novell-portlet.xml**

Portal subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\PortalService-conf**config.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\PortalService-conf**config.xml**

Project archive	Layout type	Location
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\PortalService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\PortalService-conf**config.xml**

Portlet configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\PortletService-conf**config.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\PortletService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\PortletService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\PortletService-conf**config.xml**

Rule subsystem configuration file

Project Archive	Layout Type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\RuleService-conf**config.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\RuleService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\RuleService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\RuleService-conf**config.xml**

Search subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\SearchService-conf**config.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\SearchService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\SearchService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\SearchService-conf**config.xml**

Security subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\SecurityService-conf**config.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\SecurityService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\SecurityService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\SecurityService-conf**config.xml**

User subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\UserService-conf**config.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\UserService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\UserService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\UserService-conf**config.xml**

Portal configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\conf**config.xml**
	Source	*project_name.spf**Portal_app_name*.spf\WEB-INF\conf**config.xml**
WAR	Archive	*project_name.spf*\WEB-INF\conf**config.xml**
	Source	*project_name.spf*\WEB-INF\conf**config.xml**

Workflow subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\ WorkflowService-conf**config.xml**
	Source	*project_name.spf*\library\ ConfigService\ ConfigService.spf\WorkflowService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ ConfigService.jar\WorkflowService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ ConfigService.spf\WorkflowService-conf**config.xml**

Pageflow subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\ PageflowService-conf**config.xml**
	Source	*project_name.spf*\library\ ConfigService\ ConfigService.spf\PageflowService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ ConfigService.jar\PageflowService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ ConfigService.spf\PageflowService-conf**config.xml**

WSRP Consumer subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\ WSRPConsumerService-conf**config.xml**
	Source	*project_name.spf*\library\ ConfigService\ ConfigService.spf\WSRPConsumerService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ ConfigService.jar\WSRPConsumerService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ ConfigService.spf\WSRPConsumerService-conf**config.xml**

Composer subsystem configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\ ComposerService-conf**config.xml**
	Source	*project_name.spf*\library\ ConfigService\ ConfigService.spf\ComposerService-conf**config.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ ConfigService.jar\ComposerService-conf**config.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ ConfigService.spf\ComposerService-conf**config.xml**

Services files

Content Management subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name*.spf\library\ConfigService.jar\ ContentMgmtService-conf**services.xml**
	Source	*project_name*.spf\library\ConfigService\ ConfigService.spf\ContentMgmtService-conf**services.xml**
WAR	Archive	*project_name*.spf\WEB_INF\lib\ ConfigService.jar/ContentMgmtService-conf**services.xml**
	Source	*project_name*.spf\WEB_INF\lib\ConfigService\ ConfigService.spf\ContentMgmtService-conf**services.xml**

Directory subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\ DirectoryService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ ConfigService.spf\DirectoryService-conf**services.xml**

Project archive	Layout type	Location
WAR	Archive	*project_name.spf*\WEB_INF\lib\ ConfigService.jar\DirectoryService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ ConfigService.spf\DirectoryService-conf**services.xml**

Framework subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\FrameworkService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\FrameworkService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\FrameworkService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\libConfigService\ConfigService.spf\FrameworkService-conf**services.xml**

Portal subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\PortalService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\PortalService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\PortalService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\PortalService-conf**services.xml**

Portlet services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\PortletService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\PortletService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\PortletService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\PortletService-conf**services.xml**

Resource set configuration file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\conf**resourceset.xml**
	Source	*project_name.spf**Portal_app_name*.spf\WEB-INF\conf**resourceset.xml**
WAR	Archive	*project_name.spf*\WEB-INF\conf**resourceset.xml**
	Source	*project_name.spf*\WEB-INF\conf**resourceset.xml**

Rule subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\RuleService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\RuleService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\RuleService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\RuleService-conf**services.xml**

Search subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\SearchService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\SearchService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\SearchService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\SearchService-conf**services.xml**

Security subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\SecurityService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\SecurityService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\SecurityService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\SecurityService-conf**services.xml**

User subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\UserService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\UserService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\UserService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\UserService-conf**services.xml**

Portal services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_App_name*.war\WEB-INF\conf**services.xml**
	Source	*project_name.spf**Portal_App_name*.spf\WEB-INF\conf**services.xml**
WAR	Archive	*project_name.spf*\WEB-INF\conf**services.xml**
	Source	*project_name.spf*\WEB-INF\conf**services.xml**

Workflow subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\WorkflowService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\WorkflowService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\WorkflowService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\WorkflowService-conf**services.xml**

Pageflow subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\PageflowService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\PageflowService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\PageflowService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\PageflowService-conf**services.xml**

WSRP Consumer subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\WSRPConsumerService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\WSRPConsumerService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\WSRPConsumerService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\WSRPConsumerService-conf**services.xml**

Composer subsystem services file

Project archive	Layout type	Location
EAR	Archive	*project_name.spf*\library\ConfigService.jar\ComposerService-conf**services.xml**
	Source	*project_name.spf*\library\ConfigService\ConfigService.spf\WSRPComposerService-conf**services.xml**
WAR	Archive	*project_name.spf*\WEB_INF\lib\ConfigService.jar\WSRPComposerService-conf**services.xml**
	Source	*project_name.spf*\WEB_INF\lib\ConfigService\ConfigService.spf\WSRPComposerService-conf**services.xml**

Resource set descriptors

Framework database descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**framework-database***mydbdescriptor*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**framework-database***mydbdescriptor*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**framework-database***mydbdescriptor*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**framework-database***mydbdescriptor*.xml

Views descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**my-views***myview*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**my-views***myview*.xml

Project archive	Layout type	Location
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**my-views***myview*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**my-views***myview*.xml

Pageflow process descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**pageflow-process**\mypageflowprocess*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**pageflow-process**\mypageflowprocess*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**pageflow-process**\mypageflowprocess*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**pageflow-process**\mypageflowprocess*.xml

Portal category descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-category**\myportalcategory*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-category**\myportalcategory*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-category**\myportalcategory*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-category**\myportalcategory*.xml

Portal component descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-component**\myportalcomponent*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-component**\myportalcomponent*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-component**\myportalcomponent*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-component**\myportalcomponent*.xml

Portal data definition descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-data-definition**\mydatadefinition*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-data-definition**\mydatadefinition*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-data-definition**\mydatadefinition*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-data-definition**\mydatadefinition*.xml

Portal device profile descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-device-profile**\mydeviceprofile*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-device-profile**\mydeviceprofile*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-device-profile**\mydeviceprofile*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-device-profile**\mydeviceprofile*.xml

Portal Layout descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-layout** \mylayout*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-layout**\mylayout*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-layout**\mylayout*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-layout**\mylayout*.xml

Portal option descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-option**\myportaloption*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-option**\myportaloption*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-option**\myportaloption*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-option**\myportaloption*.xml

Portal page descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-page**\myportalpage*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-page**\myportalpage*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-page**\myportalpage*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-page**\myportalpage*.xml

Portal portlet fragment deployment descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-portlet**\myportlet*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-portlet**\myportlet*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-portlet**\myportlet*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-portlet**\myportlet*.xml

Portal style descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-style**\myportalstyle*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-style**\myportalstyle*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-style**\myportalstyle*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-style**\myportalstyle*.xml

Portal theme descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**portal-theme**\myportaltheme*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-theme**\myportaltheme*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**portal-theme**\myportaltheme*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**portal-theme**\myportaltheme*.xml

Rule descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**rule**\myrule*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule**\myrule*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**rule**\myrule*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule**\myrule*.xml

Rule action macro descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**rule-action-macro**\myactionmacro*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-action-macro**\myactionmacro*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**rule-action-macro**\myactionmacro*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-action-macro**\myactionmacro*.xml

Rule condition macro descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**rule-condition-macro**\myconditionmacro*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-condition-macro**\myconditionmacro*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**rule-condition-macro**\myconditionmacro*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-condition-macro**\myconditionmacro*.xml

Rule group binding descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**rule-group-binding**\mygroupbinding*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-group-binding**\mygroupbinding*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**rule-group-binding**\mygroupbinding*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-group-binding**\mygroupbinding*.xml

Rule pipeline descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**rule-pipeline***mypipeline*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-pipeline***mypipeline*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**rule-pipeline***mypipeline*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-pipeline***mypipeline*.xml

Rule pipeline binding descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**rule-pipeline-binding***mypipelinebinding*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-pipeline-binding***mypipelinebinding*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**rule-pipeline-binding***mypipelinebinding*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-pipeline-binding***mypipelinebinding*.xml

Rule user binding descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**rule-user-binding***myuserbinding*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-user-binding***myuserbinding*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**rule-user-binding***myuserbinding*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**rule-user-binding***myuserbinding*.xml

Security role descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**security-role***mysecurityrole*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**security-role***mysecurityrole*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**security-role***mysecurityrole*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**security-role***mysecurityrole*.xml

Workflow activity policy descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**workflow-activity-policy****activity-policy.xml**
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**workflow-activity-policy****activity-policy.xml**
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**workflow-activity-policy****activity-policy.xml**
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**workflow-activity-policy****activity-policy.xml**

Workflow process descriptor

Project archive	Layout type	Location
EAR	Archive	*project_name.spf**Portal_app_name*.war\WEB-INF\lib*Portal_app_name*_resource.jar**workflow-process***myworkflowprocess*.xml
	Source	*project_name.spf**Portal_app_name*.spf*Portal_app_name*-resource.spf\data**workflow-process***myworkflowprocess*.xml
WAR	Archive	*Portal_app_name*.spf\WEB-INF\lib*Portal_app_name*_resource.jar**workflow-process***myworkflowprocess*.xml
	Source	*Portal_app_name*.spf*Portal_app_name*-resource.spf\data**workflow-process***myworkflowprocess*.xml

Portal application resources

This section describes how to locate portal resources by searching your exteNd Director project resource set.

➢ **To locate a portal resource:**

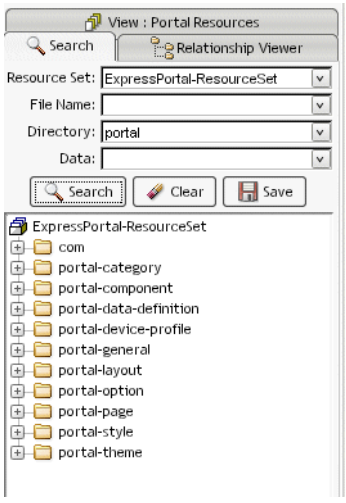
1 In the exteNd Director development environment Navigation Pane, click the **Resources** tab.

2 Click the **Search** tab.

The Search tab displays a panel for specifying searches:

3 In the Directory field, enter **portal** and click **Search.**

This displays the directories containing portal resources—for example:



4 Select the appropriate directory:

Portal resource directory	Contents
portal-category	Portal category descriptors (xml)

Portal resource directory	Contents
portal-component	Portal component descriptors (xml)
portal-data-definition	Transcoding definition for Phonelist portlet (xml)
portal-device-profile	Device profile descriptors for supported wireless devices (xml)
portal-general	Various portal artifacts such as image files and Java script.
portal-layout	Portal layout definitions and layout descriptors (xml)
portal-option	Portal option descriptors (xml)
portal-page	Portal PID pages (html and xhtml)
portal-portlet	Portlet descriptors (xml)
portal-style	Portal style sheets (xsl and xml)
portal-theme	Style sheets and descriptors for each portal theme (css and xml).

TIP: To create a writable version of a resource, select it, right-click, and choose **Save as**.

Index

E

F

G

H

O

P

R

S

T

U

V

W

X