Service Registry 3.1 Administration Guide

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

Part No: 819–4640–10
February 2007

070226@16599

Contents

Preface

The *Service Registry 3.1 Administration Guide* describes how to configure Service Registry ("the Registry") after installation and how to use the administration tool provided with the Registry. This book also describes other administrative tasks, such as backing up and restoring the Registry database.

Service Registry is an ebXML Registry: a federated registry and repository that manages all types of electronic content described by standard and extensible metadata. It provides federated, secure information management of Service Oriented Architecture (SOA) and other content and metadata. It supports the ebXML Registry 3.0 and UDDI 3.0 registry protocols.

Who Should Use This Book

The *Administration Guide* is intended for those who need to install, uninstall, and administer the Registry, as well as for those who want to create content for the Registry in bulk rather than use the Web Console to do so.

You should be familiar with the basics of a UNIX® command shell environment on your operating system (the Solaris™ Operating System, Linux, or HP-UX).

Before You Read This Book

Before you read this book, you must install the Registry as described in *Sun Java Enterprise System 5 Installation Guide for UNIX*.

Service Registry is a component of Sun Java Enterprise System ("Java ES"), a software infrastructure that supports enterprise applications distributed across a network or Internet environment. You should be familiar with the Java ES documentation at `http://docs.sun.com/coll/1286.2`.

Some administrative tasks require you to be familiar with the basic concepts of these specifications:

- *ebXML Registry Information Model Version 3.0*
- *ebXML Registry Services and Protocols Version 3.0*

How This Book Is Organized

The contents of this book are as follows:

Chapter 1 describes how to configure Service Registry after you install it and how to perform other administrative tasks.

Chapter 2 describes the use of the administration tool.

Service Registry Documentation Set

The Service Registry documentation set is available at `http://docs.sun.com/coll/1314.2`. To learn about Service Registry, refer to the books listed in the following table.

TABLE P–1 Service Registry Documentation

Document Title	Contents
Service Registry 3.1 Release Notes	Contains the latest information about Service Registry, including known problems.
Service Registry 3.1 Administration Guide	Describes how to configure Service Registry after installation and how to use the administration tool provided with the Registry. It also describes how to perform other administrative tasks.
Service Registry 3.1 User's Guide	Describes how to use the Service Registry Web Console to search Service Registry and to publish data to it.
Service Registry 3.1 Developer's Guide	Describes how to use the Java API for XML Registries (JAXR) to search Service Registry and to publish data to it.

Related Books

When you install Service Registry, it is deployed to the Sun Java System Application Server. For information about administering Application Server, refer to *Sun Java System Application Server Enterprise Edition 8.2 Administration Guide*.

The Java ES documentation set describes deployment planning and system installation. The URL for system documentation is `http://docs.sun.com/coll/1286.2`. For an introduction to Java ES, refer to the books in the order in which they are listed in the following table.

TABLE P–2 Java Enterprise System Documentation

Document Title	Contents
Sun Java Enterprise System 5 Release Notes for UNIX *Sun Java Enterprise System 5 Release Notes for Microsoft Windows*	Contains the latest information about Java ES, including known problems. In addition, components have their own release notes listed in the Release Notes Collection (`http://docs.sun.com/coll/1315.2`).
Sun Java Enterprise System 5 Technical Overview	Introduces the technical and conceptual foundations of Java ES. Describes components, the architecture, processes, and features.
Sun Java Enterprise System Deployment Planning Guide	Provides an introduction to planning and designing enterprise deployment solutions based on Java ES. Presents basic concepts and principles of deployment planning and design, discusses the solution life cycle, and provides high-level examples and strategies to use when planning solutions based on Java ES.
Sun Java Enterprise System 5 Installation Planning Guide	Helps you develop the implementation specifications for the hardware, operating system, and network aspects of your Java ES deployment. Describes issues such as component dependencies to address in your installation and configuration plan.
Sun Java Enterprise System 5 Installation Guide for UNIX *Sun Java Enterprise System 5 Installation Guide for Microsoft Windows*	Guides you through the process of installing Java ES. Also shows how to configure components after installation, and verify that they function properly.
Sun Java Enterprise System 5 Installation Reference for UNIX	Gives additional information about configuration parameters, provides worksheets to use in your configuration planning, and lists reference material such as default directories and port numbers on the Solaris Operating System and Linux operating environment.
Sun Java Enterprise System 5 Upgrade Guide for UNIX *Sun Java Enterprise System 5 Upgrade Guide for Microsoft Windows*	Provides instructions for upgrading to Java ES 5 from previously installed versions.
Sun Java Enterprise System 5 Monitoring Guide	Gives instructions for setting up the Monitoring Framework for each product component and using the Monitoring Console to view real-time data and create monitoring rules.
Sun Java Enterprise System Glossary	Defines terms that are used in Java ES documentation.

The URL for all documentation about Java ES and its components is `http://docs.sun.com/prod/entsys.5`.

Default Paths and File Names

The following table describes the default paths and file names that are used in this book.

TABLE P–3 Default Paths and File Names

Placeholder	Description	Default Value
ServiceRegistry-base	Represents the base installation directory for Service Registry.	Solaris OS: `/opt/SUNWsrvc-registry` Linux and HP-UX systems: `/opt/sun/srvc-registry`
RegistryDomain-base	Represents the directory where the Application Server domain for Service Registry is located and where the Service Registry database is located.	Solaris OS: `/var/opt/SUNWsrvc-registry` Linux and HP-UX systems: `/var/opt/sun/srvc-registry`
Ant-base	Represents the directory where the Java ES version of the Ant tool is located.	Solaris OS: `/usr/sfw/bin` Linux and HP-UX systems: `/opt/sun/share/bin`

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P–4 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	A placeholder to be replaced with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file.

Shell Prompts in Command Examples

The following table shows default system prompts and superuser prompts.

TABLE P–5 Shell Prompts

Shell	Prompt
C shell on UNIX and Linux systems	`machine_name%`
C shell superuser on UNIX and Linux systems	`machine_name#`
Bourne shell and Korn shell on UNIX and Linux systems	`$`
Bourne shell and Korn shell superuser on UNIX and Linux systems	`#`
Microsoft Windows command line	`C:\`

Symbol Conventions

The following table explains symbols that might be used in this book.

TABLE P–6 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional arguments and command options.	`ls [-l]`	The `-l` option is not required.
`{ | }`	Contains a set of choices for a required command option.	`-d {y|n}`	The `-d` option requires that you use either the `y` argument or the `n` argument.
`${ }`	Indicates a variable reference.	`${com.sun.javaRoot}`	References the value of the `com.sun.javaRoot` variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (http://www.sun.com/documentation/)
- Support (http://www.sun.com/support/)
- Training (http://www.sun.com/training/)

Searching Sun Product Documentation

Besides searching Sun product documentation from the docs.sun.com℠ web site, you can use a search engine by typing the following syntax in the search field:

search-term `site:docs.sun.com`

For example, to search for "broker," type the following:

```
broker site:docs.sun.com
```

To include other Sun web sites in your search (for example, java.sun.com, www.sun.com, and developers.sun.com), use `sun.com` in place of `docs.sun.com` in the search field.

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to http://docs.sun.com and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 819-4640.

◆ ◆ ◆ CHAPTER 1

Configuring and Setting Up Service Registry

This chapter describes how to configure Service Registry after you install it and how to perform other administrative tasks.

This chapter contains the following sections:

Configuring Service Registry

The *Sun Java Enterprise System 5 Installation Guide for UNIX* describes how to perform post-install configuration of Service Registry using default property settings for the Registry. To use custom property settings, make a copy of the file *ServiceRegistry-base*`/install/install.properties` or *ServiceRegistry-base*`/install/install.properties.template` and edit it before you perform the configuration.

To configure the Registry, you can be logged in as `root` or become superuser, or you can be logged in as an non-root user.

For security reasons, it is recommended that you configure the Registry as a non-root user. For instructions, see "To Configure Service Registry as a Non-Root User Using Custom Properties After a Configure Later Installation" on page 17.

The *ServiceRegistry-base* location is `/opt/SUNWsrvc-registry` on Solaris OS and `/opt/sun/srvc-registry` on Linux and HP-UX systems.

Note – Before you configure Service Registry, you must first install and configure Sun Java System Application Server ("Application Server"). The configuration process for Service Registry installs the Registry into an Application Server domain.

It is recommended that you install Application Server in its default location. If you installed Application Server in a non-default location, follow the instructions in "Configuring Service Registry for a Non-Default Application Server Installation" on page 22 before you configure Service Registry.

The `install.properties` file contains a set of modifiable property settings. The properties that are listed in Table 1–1 are used by the configuration process. Each property name has the prefix `registry.install.` (terminating in a period). Some of these properties set non-default ports for the Application Server domain created for the Registry. Others set configuration properties described elsewhere in this manual.

You can configure multiple instances of Service Registry either as root or as a non-root user. If you do so, you must change the `registry.install.dataHome` property to be unique for each Registry instance. If you want to be able to run more than one instance at a time on the server, you must also change the eight port properties to be unique for each instance. For multiple non-root instances, you may want to change the value of `registry.install.CACertDir` to be unique for each instance.

TABLE 1–1 Service Registry Configuration Properties

Property Name	Description	Default Property Value
`DomainName`	Application Server domain name	`registry`
`ServerInstanceHost`	The default hostname where Service Registry can be accessed	`localhost`
`ServerInstancePort`	Application Server HTTP port for Service Registry	`6480`
`ServerInstanceSecurePort`	Application Server HTTPS port for Service Registry	`6443`
`ServerJMSPort`	Application Server Message Queue port for Service Registry	`6484`

TABLE 1–1 Service Registry Configuration Properties *(Continued)*

Property Name	Description	Default Property Value
`ServerIIOPPort`	Application Server IIOP port for Service Registry	`6485`
`ServerIIOPSecurePort`	Application Server IIOP secure port for Service Registry	`6486`
`ServerIIOPMutualAuthPort`	Application Server IIOP mutual authentication port for Service Registry	`6487`
`AdministrationJMXPort`	Application Server JMX port for Service Registry	`6488`
`AdministrationPort`	Application Server Administrative Server port for Service Registry	`6489`
`AdministratorUserID`	User name used to access Application Server Administrative Server	`admin`
`AdministratorPassword`	Password used to access Application Server Administrative Server	`12345678` when you configure as `root` None when you configure as a non-root user
`ApplicationServerKeystorePassword`	Password used to access Application Server keystore	`12345678` when you configure as `root` None when you configure as a non-root user
`RegistryServerKeystorePassword`	Password used to access Service Registry keystore	`12345678` when you configure as `root` None when you configure as a non-root user
`clientDatabase`	Determines whether the Service Registry database runs in embedded or Network Server mode (`true` for Network Server mode)	`false`
`RequireDatabaseAuthentication`	Determines whether access to the Service Registry database requires user authentication	`false`

TABLE 1–1 Service Registry Configuration Properties *(Continued)*

Property Name	Description	Default Property Value
`DatabaseUserID`	User ID for accessing the Service Registry database if user authentication is required	`APP`
`DatabasePassword`	Password for accessing the Service Registry database if user authentication is required	`app123` when you configure as `root` None when you configure as a non-root user
`backupDir`	Directory to be used for Service Registry backups. Normally commented out; remove comment to specify an alternate location.	*RegistryDomain-base*`/3.0/backup` when you configure as `root` `$HOME/srvc-registry/3.0/backup` when you configure as a non-root user
`dataHome`	Directory where Service Registry data is stored	*RegistryDomain-base* when you configure as `root` `$HOME/srvc-registry` when you configure as a non-root user
`CACertDir`	Directory for added certificates that the Application Server domain should trust	*ServiceRegistry-base*`/install/cacerts` when you configure as `root` `$HOME/srvc-registry/cacerts` when you configure as a non-root user

▼ To Configure Service Registry as Root Using Custom Properties After a Configure Later Installation

Before You Begin These instructions assume that you are logged in as `root` or have become superuser.

1. **Change to the** *ServiceRegistry-base*`/install` **directory.**

2. **Copy the file** `install.properties` **to a secure location.**

   Use a command like the following:

   ```
   cp install.properties $HOME/hidden_dir/sr.properties
   ```

3. **Change the permissions on the properties file to make it writable.**

   Use a command like the following:

   ```
   chmod 600 $HOME/hidden_dir/sr.properties
   ```

4. **Edit the modifiable properties in the file.**

   For example, it is recommended that you change all the passwords from the default values.

5. **Change the permissions back to the original read-only value.**

   Use a command like the following:

   ```
   chmod 400 $HOME/hidden_dir/sr.properties
   ```

6. **In the *ServiceRegistry-base*/`install` directory, run the following command (all on one line), specifying the location of the modified `install.properties` file.**

   Use a command like the following:

   Ant-base`/ant -f build-install.xml -Dinstall.properties=$HOME/hidden_dir/sr.properties configure`

   The `ant` command requires the `JAVA_HOME` environment variable to be set. Ordinarily, you set this variable to the following value:

   ```
   /usr/jdk/entsys-j2se
   ```

   The Registry configuration process creates an Application Server domain at *RegistryDomain-base*`/domains/${registry.install.DomainName}`. The default domain name is `registry`. The configuration process then starts the domain, deploys the Registry, and leaves the domain running.

   The Registry configuration process installs the Registry database and server keystore in the directory *RegistryDomain-base*`/3.0`. This directory is not removed when the Registry is uninstalled, so that the database can be preserved for use in a future release. The administrator has control over when and whether to remove this directory.

   The *RegistryDomain-base* location is `/var/opt/SUNWsrvc-registry` on Solaris OS and `/var/opt/sun/srvc-registry` on Linux and HP-UX systems.

   The Registry configuration process creates a directory named *ServiceRegistry-base*`/install/cacerts` for you to place added certificates that the Application Server domain should trust.

7. **Review the output of the `ant configure` command for any errors.**

   If there are no errors, you can now begin using the Web Console or the Admin Tool.

▼ To Configure Service Registry as a Non-Root User Using Custom Properties After a Configure Later Installation

Before You Begin These instructions assume that you are logged in as an ordinary user, not as `root`, and that someone logged in as `root` has installed Service Registry.

1. **Change to the** *ServiceRegistry-base*`/install` **directory.**

2. **Copy the file** `install.properties.template` **to a location in your home directory and rename it.**

   Use a command like the following:

   ```
   cp install.properties.template $HOME/sr.properties
   ```

3. **Open your copy of the file in a text editor and modify properties as needed.**

   You must supply password values for the following properties, which are left empty in the template file:

   ```
   registry.install.AdministratorPassword=
   registry.install.ApplicationServerKeystorePassword=
   registry.install.RegistryServerKeystorePassword=
   ```

4. **Change the permissions on the file so that only you can read it.**

   Use a command like the following:

   ```
   chmod 400 $HOME/sr.properties
   ```

5. **In the** *ServiceRegistry-base*`/install` **directory, run a command like the following (all on one line), specifying the location of the modified file.**

   Ant-base`/ant -f build-install.xml -Dinstall.properties=$HOME/sr.properties configure`

   Use a similar command to run any other configuration targets that you need.

   The Registry configuration process creates an Application Server domain at `$HOME/srvc-registry/domains/${registry.install.DomainName}`. The default domain name is `registry`.

   The Registry configuration process installs the Registry database and server keystore in the directory `$HOME/srvc-registry/3.0`.

   The Registry configuration process creates a directory named `$HOME/srvc-registry/cacerts` for you to place added certificates that the Application Server domain should trust.

6. **Review the output of the** `ant configure` **command for any errors.**

   If there are no errors, you can now begin using the Web Console or the Admin Tool.

Administering the Application Server Domain for Service Registry

The configuration process for Service Registry by default creates an Application Server domain named `registry`, to which the Service Registry web application is deployed. This domain is in the *RegistryDomain-base*`/domains/registry` directory.

This location is different from the default location for Application Server domains, `/var/opt/SUNWappserver/domains` (Solaris OS) or `/var/opt/sun/appserver/domains` (Linux and HP-UX systems).

Note – It is recommended that you not run any applications other than Service Registry in the `registry` domain.

To administer the `registry` domain, you can use the Application Server Administration Console ("Admin Console"). You can use the Admin Console to start and stop the domain, view the server log, and perform other administrative tasks. See "To Use the Application Server Admin Console" on page 20 for details.

You can also examine the server log directly. The log is in the file *RegistryDomain-base*`/domains/registry/logs/server.log`.

In addition to the Admin Console, you can use the `asadmin` command to administer the `registry` domain. Because the domain is not in the default location, you must specify the `--domaindir` option when you use `asadmin` commands that provide that option. The argument to the `--domaindir` option is *RegistryDomain-base*`/domains`.

If you want to use the `--passwordfile` option of `asadmin` commands, you need a file with a copy of the administrator password for the Registry domain. To create such a file, use the `generate.password.file` target of the `build-install.xml` file. The file is *RegistryDomain-base*`/3.0/data/security/pw.txt`.

The `registry` domain uses a set of non-default ports so as not to cause conflicts with the default Application Server domain, `domain1`. These Service Registry port values are registered with the Internet Assigned Numbers Authority (IANA). Table 1–2 lists and describes these ports. For more information, see "Ports in the Application Server" in *Sun Java System Application Server Enterprise Edition 8.2 Administration Guide*.

TABLE 1–2 Service Registry Domain Default Ports

Port Value	Description
6480	HTTP port

TABLE 1–2 Service Registry Domain Default Ports *(Continued)*

Port Value	Description
6443	HTTPS over SSL
6484	Message Queue port
6485	IIOP port
6486	IIOP SSL port
6487	IIOP Mutual Authentication port
6488	JMX port
6489	Application Server domain administration port

▼ To Use the Application Server Admin Console

1. **In a web browser, go to the URL** `https://`*hostname*`:6489/`**.**

   hostname is the system on which Application Server and Service Registry are running.

2. **Accept the certificate that is offered.**

   A login page appears.

3. **On the login page, type** `admin` **in the User Name field.**

   If you changed the default value of the `registry.install.AdministratorUserID` property when you configured the Registry, type the value you specified.

4. **Type the Application Server administrator password in the Password field. Use the value that you specified for the** `registry.install.AdministratorPassword` **property when you configured the Registry. The default is** `12345678`**.**

5. **Click Log In.**

See Also For details on using the Admin Console, refer to the online help for the Admin Console or to the *Sun Java System Application Server Enterprise Edition 8.2 Administration Guide*.

Changing the Service Registry Logging Level

To change the logging level for Service Registry, follow the instructions in the Admin Console online help. The property to specify in the Additional Properties area is `org.freebxml.omar`.

To change the logging for particular Service Registry subcomponents, refer to the following file: *RegistryDomain-base*`/domains/registry/applications/j2ee-modules/soar/WEB-INF/`

`classes/log4j.properties`. You can specify any of the subcomponent names listed in this file. Do not include the string `log4j.logger.` For example, you can log server calls by specifying `org.freebxml.omar.server`.

▼ To Stop and Restart the Application Server Domain for the Registry

The configuration process for the Registry starts the Application Server domain in which the registry is deployed. After you perform certain administrative tasks, you need to stop and restart the domain. Examples of such tasks are "Configuring the Java Virtual Machine (JVM) for the Registry Domain" on page 23 and "Creating an Administrator" on page 25.

The Admin Console informs you if you need to restart the domain. You can use the Admin Console to perform this task. If you are using the `asadmin` command, you can use Ant tasks to stop and start the domain.

1. **Change to the Service Registry install directory.**

   `cd` *ServiceRegistry-base*`/install`

2. **Run the following command (all on one line):**

   Ant-base`/ant -f build-install.xml -Dinstall.properties=`*props-file* **`appserver.domain.bounce`**

   where *props-file* is the path name of the copy of `install.properties` file that you edited in "To Configure Service Registry as Root Using Custom Properties After a Configure Later Installation" on page 16 or "To Configure Service Registry as a Non-Root User Using Custom Properties After a Configure Later Installation" on page 17.

   The `appserver.domain.bounce` target stops the domain and then restarts it.

   The `build-install.xml` file also contains separate Ant targets for stopping and starting the Registry domain. To stop the domain, use the Ant target `appserver.domain.stop`. To start the domain, use the Ant target `appserver.domain.start`.

▼ To Add Root Certificates to the Trusted Certificates in the Registry Domain

This task extends the list of trusted certificates in the Application Server `registry` domain.

Perform this task only if you use a third-party certificate and the root Certificate Authority (CA) certificate for the third party is not already in the Application Server truststore. Do not perform this task if you use only registry-issued certificates.

To determine whether the CA certificate you need is already available, you can use the `list.cacerts` target of the `build-install.xml file`:

Ant-base`/ant -f build-install.xml -Dinstall.properties=`*props-file* **`list.cacerts`**

where *props-file* is the path name of the copy of `install.properties` file that you edited in "To Configure Service Registry as Root Using Custom Properties After a Configure Later Installation" on page 16 or "To Configure Service Registry as a Non-Root User Using Custom Properties After a Configure Later Installation" on page 17.

1 **Download any root certificates that you want to support.**

Sites that provide root certificates include the following:

- `http://www.entrust.net/developer/`
- `http://www.geotrust.com/resources/root_certificates/`
- `http://www.thawte.com/roots/`
- `http://www.verisign.com/support/roots.html`

2 **If necessary, use the `unzip` command to extract `.cer` files from the downloaded archive.**

Note – Some files have the suffix `.der`.

---

3 **Copy the `.cer` files to the directory specified by the `registry.install.CACertDir` property in your copy of the `install.properties` file.**

This value is normally *ServiceRegistry-base*`/install/cacerts` if you configured as `root`, or `$HOME/srvc-registry/cacerts` if you configured as a non-root user.

4 **Change to the directory *ServiceRegistry-base*`/install`.**

5 **Run the following command (all on one line):**

Ant-base`/ant -f build-install.xml -Dinstall.properties=`*props-file* `install.cacerts`

This command installs any certificates found in the directory specified by the `registry.install.CACertDir` property into the Application Server domain truststore.

You can use the `list.cacerts` target again to make sure that the certificates have been installed correctly.

6 **Follow the instructions in "To Stop and Restart the Application Server Domain for the Registry" on page 21.**

Configuring Service Registry for a Non-Default Application Server Installation

The default location for installing Application Server is `/opt/SUNWappserver/appserver` on Solaris OS and `/opt/sun/appserver` on Linux and HP-UX systems. If you installed Application Server in a different location, you must edit a copy of the file `install.properties` before you configure Service Registry.

▼ To Edit a Copy of the `install.properties` File

1. **Change to the *ServiceRegistry-base*`/install` directory.**
2. **Copy the file `install.properties` to a secure location and rename it.**
   Use a command like the following:
   ```
   cp install.properties $HOME/hidden_dir/sr.properties
   ```
3. **Change the permissions on the properties file to make it writable.**
   Use a command like the following:
   ```
   chmod 600 $HOME/hidden_dir/sr.properties
   ```
4. **Open the file in a text editor.**
5. **Find the commented-out definition of the property `appserver.root.dir`.**
6. **Remove the comment character (#) and replace the property definition with the actual location of Application Server.**
7. **Save and close the file.**

Next Steps Continue with the instructions in "Configuring Service Registry" on page 13.

Configuring the Java Virtual Machine (JVM) for the Registry Domain

Service Registry requires the following in order to work correctly:

- The Registry must be able to access external web sites
- The Application Server domain for the Registry must have enough memory available to it

Any registry object can have an `ExternalLink` object, which specifies an external URL associated with that registry object. Any `Service` object can have a `ServiceBinding`, which can also point to an external URL. In order for users to create `ExternalLink` and `ServiceBinding` objects, Service Registry must be able to validate the URL, and this task requires access to external web sites. If the Registry is deployed behind a firewall, you need to set a proxy configuration that allows this access.

Proxy configuration requires you to specify a web proxy host and port as Java Virtual Machine (JVM) options of the Application Server domain where Service Registry is deployed.

It is also possible for the Registry to run out of memory. To prevent this problem from occurring, configure a JVM option to increase the memory available to the Application Server domain for the Registry.

Perform the following task to configure JVM options for the Registry.

▼ To Configure JVM Options for the Registry Domain

1. **Log in to the Application Server Admin Console as described in "To Use the Application Server Admin Console" on page 20.**
2. **Expand the Configurations node.**
3. **Expand the server node,** `server-config (Admin Config)`**.**
4. **Click JVM Settings.**
5. **Click the JVM Options tab.**
6. **Click Add JVM Option.**
7. **In the text field, type the following (all on one line):**

`-Dhttp.proxyHost=`*hostname.domainname* **`-Dhttp.proxyPort=8080 -Dhttp.nonProxyHosts=localhost`**

   The port value is usually 8080. If the port is different in your location, specify the correct value.

8. **Click Add JVM Option again.**
9. **In the text field, type the following (all on one line):**

   `-XX:MaxPermSize=128m`

10. **Click Save.**
11. **Follow the instructions in "To Stop and Restart the Application Server Domain for the Registry" on page 21.**

Creating an Administrator

The Service Registry administration tool enables some tasks that only a user who is registered as an administrator can perform. In addition, an administrator might be called upon to implement life cycle changes (for example, approvals) to objects other users submit.

An administrator can also change the default access control policy (ACP). However, writing an ACP is currently a manual process that requires knowledge of OASIS eXtensible Access Control Markup Language (XACML). For details, refer to Chapter 9, "Access Control Information Model," of ebXML RIM 3.0, especially the examples in Sections 9.7.6 through 9.7.8. See "Before You Read This Book" on page 7 for information on how to find the ebXML RIM 3.0 specification.

▼ To Create an Administrator

To register yourself as an administrator, follow these steps:

1. **Either perform user registration as described in "Creating a User Account" in** *Service Registry 3.1 User's Guide*, **or add yourself as a user by using the** `add user` **of the Admin Tool, described in** **"`add user`" on page 49****.**

2. **If you used the Web Console to register, obtain the unique identifier of your** `User` **object as follows:**

   a. **Use the Web Console to perform a Basic Query, with the Object Type set to User.**

   b. **Click the Details link to view the** `User` **object the Registry created for you.**

   c. **Write down the Unique Identifier field value, or copy and paste it into a file.**

   If you used the `add user` command, use the `users` command to get a list of users, then copy the identifier value for your user name.

3. **Change to the directory**
   RegistryDomain-base`/domains/registry/applications/j2ee-modules/soar/WEB-INF/classes`**.**

4. **Open the file** `omar.properties` **in a text editor.**

5. **Find the definition of the property**
   `omar.security.authorization.registryAdministrators`**.**

6 **Edit the property definition by adding a vertical bar (`|`), followed by the unique identifier string that you copied in Step 2.**

The property definition must all be on one line and must not contain spaces. After you finish, it will look something like this (all on one line):

```
omar.security.authorization.registryAdministrators=
urn:freebxml:registry:predefinedusers:registryoperator|
urn:uuid:77f5c196-79de-4286-8483-8d80def3583b
```

7 **Save and close the `omar.properties` file.**

8 **Follow the instructions in "To Stop and Restart the Application Server Domain for the Registry" on page 21.**

Next Steps To create additional administrators, you do not have to edit the `omar.properties` file. You can use either the Admin Tool or the Web Console to add users, and you can use the Web Console to classify the users as administrators.

Specifying Who Can Perform User Registration

By default, anyone who can access the Service Registry URL is permitted to perform user registration by using the User Registration Wizard. Any registered user can publish content to the Registry.

By defining the property `omar.security.selfRegistration.acl` in the file `omar.properties`, you can restrict this capability so that only specified people can perform user registration.

After you complete this task, the only people who can register by using the Wizard will be those who have the first and last names specified by the property definition. Other people who try to register will see an error message when they click the Next button after filling out the User Authentication Details form in Step 3 of the User Registration Wizard. The error message says `User registration failed.` After the message is a line that reports a `UserNotFoundException`.

If the `omar.security.selfRegistration.acl` property is not present in `omar.properties`, or if it is defined as the empty string, any registered user can publish content to the registry.

▼ To Restrict User Registration

1 **Change to the directory** *RegistryDomain-base*`/domains/registry/applications/j2ee-modules/soar/WEB-INF/classes`**.**

2 **Open the file `omar.properties` in a text editor.**

3. **Add a definition of the following property:**

   `omar.security.selfRegistration.acl`

   You can place this property definition anywhere in the file. A logical place for the property definition is in an area where other properties with the prefix `omar.security` are defined.

   Define the property value as a comma-separated list of first and last names of people who are authorized to perform user registration, as in the following example:

   ```
   omar.security.selfRegistration.acl=Vijay Patel, Jane Doe,
   ```

4. **Save and close the `omar.properties` file.**

5. **Follow the instructions in "To Stop and Restart the Application Server Domain for the Registry" on page 21.**

Enabling Versioning of Registry Objects

By default, versioning of registry objects is disabled. If versioning is enabled, a new version of an object is created whenever any of the object's attributes is changed. Enabling versioning is an administrative task.

▼ To Enable Versioning of Registry Objects

1. **Change to the directory**
   RegistryDomain-base`/domains/registry/applications/j2ee-modules/soar/WEB-INF/classes`**.**

2. **Open the file `omar.properties` in a text editor.**

3. **Find the definition of the property**
   `omar.server.lcm.VersionManager.versionableClassList`**.**

   By default, this property has no value:

   ```
   omar.server.lcm.VersionManager.versionableClassList=
   ```

4. **Specify any objects for which you want the Registry to create new versions when the objects are modified. Separate the objects with a vertical bar (`|`).**

   For example:

   ```
   omar.server.lcm.VersionManager.versionableClassList=Service|Organization
   ```

   Use the commented-out copy of the property setting in the file as an example.

5. **Save and close the `omar.properties` file.**

6 **Follow the instructions in "To Stop and Restart the Application Server Domain for the Registry" on page 21.**

Disabling the WSDL Cataloger

By default, when a WSDL file is published to the Registry as an `ExtrinsicObject` object, the Registry creates metadata for the object using the capability defined by the ebXML Registry Profile for Web Services. It is possible to disable this capability.

▼ To Disable the WSDL Cataloger

1 **Change to the directory** *RegistryDomain-base*`/domains/registry/applications/j2ee-modules/soar/WEB-INF/classes`**.**

2 **Open the file** `omar.properties` **in a text editor.**

3 **Find the definition of the property** `omar.server.cms.classMap.urn\:oasis\:names\:tc\:ebxml-regrep\:profiles\:ws\:wsdl\:cataloging\:Service\:default`**.**

4 **Comment out this property definition by placing a pound sign (#) before the definition.**

5 **Save and close the** `omar.properties` **file.**

6 **Follow the instructions in "To Stop and Restart the Application Server Domain for the Registry" on page 21.**

Configuring the Web Console

As an administrator, you can customize some aspects of the Web Console display by editing configuration files.

This section describes the following tasks:

For information about using the Web Console, see the *Service Registry 3.1 User's Guide*.

Adding Predefined Queries

Service Registry includes several predefined queries, which appear in the Web Console Search form in the Select Predefined Query drop-down list. As an administrator, you can add new queries to the drop-down list that are specific to your installation of the Registry.

▼ To Add a Predefined Query

1 **Use the Web Console to publish an `AdhocQuery` object to the Registry.**

The name and description you specify for the query will appear in the drop-down list of predefined queries. In the SQL statement for the query, specify placeholders for user-supplied data by enclosing them in pairs of single quotes, as follows:

```
select * from registryobject where id = ''$lid''
```

2 **Write down the unique identifiers of the `AdhocQuery` object and of any placeholders in the SQL statement, or copy and paste them into a file.**

3 **Change to the directory** *RegistryDomain-base*`/3.0/jaxr-ebxml`**.**

4 **Open the file `registry-browser-config.xml` in a text editor.**

Make a copy of the original file so that you can back out your changes if you need to.

5 **Add an entry to the `registry-browser-config.xml` file, using the following format. Specify a `Parameter` element for each placeholder in the SQL statement.**

```
<Query>
  <AdhocQueryRef id="unique-identifier"/>
  <Parameter parameterName="$placeholder-name" datatype="string">
    <rim:Name>
      <rim:LocalizedString xml:lang="en" charset="UTF-8"
        value="parameter-name-in-en-locale"/>
      <rim:LocalizedString xml:lang="fr" charset="UTF-8"
        value="parameter-name-in-fr-locale"/>
    </rim:Name>
    <rim:Description>
      <rim:LocalizedString xml:lang="en" charset="UTF-8"
        value="parameter-description-in-en-locale"/>
      <rim:LocalizedString xml:lang="fr" charset="UTF-8"
        value="parameter-description-in-fr-locale"/>
    </rim:Description>
  </Parameter>
  ...
</Query>
```

The *unique-identifier* is the unique identifier of the `AdhocQuery` object.

The `parameterName` attribute value for each parameter must come from a placeholder in the SQL statement for the query.

The `datatype` attribute can have any of the following values:

- `string`: The parameter appears as a text field in the Search form.
- `taxonomyElement`: The parameter appears as a drop-down list in the Search form. If you specify a `taxonomyElement` data type, the Name and Description elements must be followed by a `SlotList` element that looks like this:

```
<rim:SlotList>
  <rim:Slot name="domain">
    <rim:ValueList>
      <rim:Value>
      classification-scheme-or-concept-id
      </rim:Value>
    </rim:ValueList>
  </rim:Slot>
</rim:SlotList>
```

  The *classification-scheme-or-concept-id* is the unique identifier of the classification scheme or concept whose concepts (or subconcepts) must appear in the drop-down list. You must publish the classification scheme if it does not already exist in the registry.

  The slot name must be `"domain"`.
- `boolean`: The parameter appears as a checkbox in the Search form.

If the `datatype` is `string` or `boolean`, you can also add a `defaultValue` attribute to the `Parameter` element to specify a default value to appear in the Search form.

Specify localized string values for each parameter name and description for any locales you support. The *parameter-name* in the current locale appears as the label of the parameter in the Search form.

Use the existing entries in the `registry-browser-config.xml` file as a reference.

6 **Save and close the `registry-browser-config.xml` file.**

7 **Follow the instructions in "To Stop and Restart the Application Server Domain for the Registry" on page 21.**

Changing the Default Query

The query that appears as the default in the Select Predefined Query drop-down list is Basic Query, which allows users to search for registry objects by name, description, and classification.

As an administrator, you can change this default to a query that is appropriate to your installation. For example, you might want the default query to be a new predefined query that you added to the Registry, as described in "Adding Predefined Queries" on page 29. To make this change, edit a property in a configuration file.

▼ To Change the Default Query

1. **Change to the directory** *RegistryDomain-base*`/domains/registry/applications/j2ee-modules/soar/WEB-INF/classes`**.**
2. **Open the file** `jaxr-ebxml.properties` **in a text editor.**
3. **Find the definition of the property** `jaxr-ebxml.thin.defaultQueryPanel`**. By default, this property is commented out:**

   ```
   #jaxr-ebxml.thin.defaultQueryPanel=
   ```

4. **Remove the comment character (#).**
5. **Set the value of the property by specifying the logical identifier of the query that will be the default, as in the following example:**

   ```
   jaxr-ebxml.thin.defaultQueryPanel=urn:oasis:names:tc:ebxml-regrep:query:MyQuery
   ```

6. **Save and close the** `jaxr-ebxml.properties` **file.**
7. **Follow the instructions in "To Stop and Restart the Application Server Domain for the Registry" on page 21.**

Hiding Classification Schemes

A tree structure of classification schemes appears in the following areas of the Web Console:

- The ClassificationScheme/Concept Selector window that appears when you click Select Classification Node in the Search menu area or when you need to choose a concept for some kinds of registry objects
- The Explore menu area

As an administrator, you can hide classification schemes from view if you do not want the classification schemes to be available to users of Service Registry. To hide classification schemes, define a property in a configuration file.

▼ To Hide Classification Schemes

1. **Change to the directory** *RegistryDomain-base*`/domains/registry/applications/j2ee-modules/soar/WEB-INF/classes`**.**

2. **Open the file** `jaxr-ebxml.properties` **in a text editor.**

3. **Set the property** `jaxr-ebxml.registryBrowser.ConceptsTreeModel.hiddenSchemesList` **by using the following syntax. All of the property definition must be on one line and must not contain spaces.**

   `jaxr-ebxml.registryBrowser.ConceptsTreeModel.hiddenSchemesList=`
   class-scheme-id1`|`*class-scheme-id2*`|...`

   Specify the logical identifier of each classification scheme that is to be hidden. If you specify more than one identifier, separate the identifiers with a vertical bar (`|`), as in the following example:

   ```
   jaxr-ebxml.registryBrowser.ConceptsTreeModel.hiddenSchemesList=
   urn:oasis:names:tc:ebxml-regrep:classificationScheme:StatusType|
   urn:oasis:names:tc:ebxml-regrep:profile:ws:classificationScheme:BindingType
   ```

4. **Save and close the** `jaxr-ebxml.properties` **file.**

5. **Follow the instructions in** **"To Stop and Restart the Application Server Domain for the Registry" on page 21.**

Configuring the Search Results Display

By default, the Web Console displays 25 search results at a time for each query. If the search returns more than 25 results, users can display additional pages of results. As an administrator, you can modify the number of search results that appears on each page.

By default, the Web Console displays certain columns in the search results area. For each object, it displays the object type, name, description, version, and version comment. For some object types, a non-default display is configured. For example, for a `ServiceBinding` object, the display includes the endpoint instead of the version information. As an administrator, you can add configuration information to display non-default data for object types of your choice.

To perform each of these tasks, you edit a configuration file.

To Configure the Number of Rows in the Search Results Display

1. **Change to the directory** *RegistryDomain-base*`/domains/registry/applications/j2ee-modules/soar/WEB-INF/classes`**.**
2. **Open the file** `jaxr-ebxml.properties` **in a text editor.**
3. **Find the definition of the property** `omar.client.thinbrowser.numSearchResults`**:**

   ```
   omar.client.thinbrowser.numSearchResults=25
   ```
4. **Change the value 25 to the value you prefer.**
5. **Save and close the** `jaxr-ebxml.properties` **file.**
6. **Follow the instructions in** **"To Stop and Restart the Application Server Domain for the Registry" on page 21.**

To Configure the Columns in the Search Results Area

You can configure columns in the Search Results area for object types. The columns display the attributes of the objects.

1. **Change to the directory** *RegistryDomain-base*`/3.0/jaxr-ebxml`**.**
2. **Open the file** `registry-browser-config.xml` **in a text editor.**
3. **Add an entry to the** `registry-browser-config.xml` **file, or edit an existing one. Use the following format.**

   This example configures a non-default display for `Service` objects.

   ```
   <ObjectTypeConfig
       className="org.freebxml.omar.client.xml.registry.infomodel.ServiceImpl"
       id="urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Service">
     <SearchResultsConfig>
       <SearchResultsColumn columnClass="java.lang.Object"
         columnHeader="Object Type" columnWidth="25" editable="false"
         method="getObjectType"/>
       <SearchResultsColumn columnClass="java.lang.Object"
         columnHeader="Name" columnWidth="25" editable="true" method="getName"/>
       <SearchResultsColumn columnClass="java.lang.Object"
         columnHeader="Description" columnWidth="30" editable="true"
         method="getDescription"/>
       <SearchResultsColumn columnClass="java.lang.Object"
         columnHeader="Status" columnWidth="15" method="getStatusAsString"/>
       <SearchResultsColumn columnClass="java.lang.Object"
   ```

```
        columnHeader="Version" columnWidth="5" method="getVersionName"/>
    </SearchResultsConfig>
</ObjectTypeConfig>
```

The `registry-browser-config.xml` file provides syntax for the `ObjectTypeConfig` element. Use the elements that are already in the file as examples. These elements configure the default display for registry objects as well as non-default displays for `ExternalLink`, `ExtrinsicObject`, and `ServiceBinding` objects.

The maximum number of columns that you can configure is 30.

For the `SearchResultsColumn` element:

- The `columnClass` attribute value is always `java.lang.Object`.
- The `columnHeader` attribute value is a key to a message in the Web Console resource bundle files. These files are contained in the directory *registryDomain-base*`/domains/registry/applications/j2ee-modules/soar/WEB-INF/classes/org/freebxml/omar/client/ui/thin/`. For example, if you type `Object Type` for the `columnHeader` value, the Web Console's backing beans use the `WebResourceBundle` class to look up a message with that key. If the `WebResourceBundle` fails to find the message, it converts the key to lowercase and tries again. If this fails again, it sets the message value to`"???"+key+"???"` and logs a warning message about the missing resource bundle entry. So, to add new localized `columnHeader` values, you must enter new message keys into the `ResourceBundle` files contained in this directory.
- The `columnWidth` attribute is not used by the Web Console.
- The `editable` attribute is not used by the Web Console.
- For the most part, you can deduce the method names for the `method` attribute from the class attributes in the *ebXML Registry Information Model Version 3.0* specification (see "Before You Read This Book" on page 7 for details). The `getStatusAsString` method can be found in the `RegistryObjectImpl` implementation class. (This release of Service Registry does not include API documentation, however.)

You can have no more than one `ObjectTypeConfig` element for each `omar.client.xml.registry.infomodel` class name.

4 **Save and close the `registry-browser-config.xml` file.**

5 **Follow the instructions in "To Stop and Restart the Application Server Domain for the Registry" on page 21.**

6 **To verify the reconfiguration, use the Search or Explore menu of the Web Console to display the objects whose columns you changed.**

Reinstalling Service Registry

If you need to uninstall and reinstall Service Registry, perform the following tasks before you reinstall:

- If the Registry database contains data that you want to preserve, back up the database as described in "Administering the Java DB Database" on page 36.
- Stop the Application Server domain for the Registry, then delete the domain. If you do not delete the domain, post-install configuration of the reinstalled Registry will fail.

If you need to reinstall the Service Registry database (for example, if the database becomes corrupted), follow the instructions in "To Reinstall the Service Registry Database" on page 35. You do not need to uninstall the database before you reinstall it.

▼ To Stop and Delete the Application Server Domain for the Registry

1. **Change to the** *ServiceRegistry-base*`/install` **directory.**

2. **Run the following command (all on one line):**

   Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file* `appserver.domain.delete`

   where *props-file* is the path name of the copy of `install.properties` file that you edited in "To Configure Service Registry as Root Using Custom Properties After a Configure Later Installation" on page 16 or "To Configure Service Registry as a Non-Root User Using Custom Properties After a Configure Later Installation" on page 17.

   This target stops the domain and then deletes it.

▼ To Reinstall the Service Registry Database

This task deletes the existing database contents, including the registered users, and recreates the default database.

1. **Change to the** *ServiceRegistry-base*`/install` **directory.**

2. **Run the following command (all on one line):**

   Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file* `install.db`

3 **Run the following command (all on one line):**

Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file*
`appserver.domain.stop export.registryOperatorCert install.cacerts`
`appserver.domain.start`

Administering the Java DB Database

The Registry uses the Java DB database. Java DB is a commercial release of the Apache Software Foundation's open source relational database project. The Apache project is called Derby.

By default, the database is located in the directory *RegistryDomain-base*`/3.0/data/registry/soar/`. Database backups are placed in the directory *RegistryDomain-base*`/3.0/backup/`, with a subdirectory name that contains the date of the backup (for example, `20060419-004759`).

By default, the Java DB database runs in embedded mode. This means that it runs in the same JVM as Service Registry, and it can accept a connection from only one client, Service Registry. No remote connections are possible. When Java DB runs in embedded mode, you can back up the database only when it is not running (an offline backup). For instructions, see "To Perform an Offline Backup of the Database" on page 38.

If you need to be able to back up the database while the registry domain is running (an online backup), you must run the Java DB database in Network Server mode. For instructions, see "To Switch From Embedded Mode to Network Server Mode" on page 38 and "To Perform an Online Backup of the Database" on page 40.

When it runs in Network Server mode, Java DB can accept multiple client connections in a familiar client/server configuration. For example, both Service Registry and a SQL client could concurrently communicate with Java DB. When Java DB runs in Network Server mode, it uses a database port whose default value is 1527. Clients such as Service Registry use this port to communicate with the database.

When Java DB runs in Network Server mode, Service Registry runs in Network Client mode.

A database that runs in Network Server mode must be password protected so that only authenticated clients can use it. You can also protect the database this way when it runs in embedded mode, but it is not essential to do so.

By default, the properties in the file *ServiceRegistry-base*`/install/install.properties` are set so that Java DB runs in embedded mode. Table 1–3 shows these settings.

TABLE 1–3 Default Property Settings for Java DB

Property Setting	Description
`registry.install.clientDatabase=false`	Enables embedded mode
`registry.install.RequireDatabaseAuthentication=false`	Does not require database authentication
`registry.install.DatabaseUserID=APP`	Sets user ID to `APP` (not used)
`registry.install.DatabasePassword=app123`	Sets password to `app123` (not used)

For more details on Java DB, consult the Java DB documentation at the Java DB web site.

▼ To Require Database Authentication

By default, database authentication is not required. It is possible to require authentication when the database is running in embedded mode, and it is necessary to require authentication when the database is running in Network Server mode.

1. **Change to the Service Registry install directory:**

   `cd` *ServiceRegistry-base*`/install`

2. **Open your copy of the `install.properties` file in a text editor.**

3. **Change the setting of the `registry.install.RequireDatabaseAuthentication` property from `false` to `true`.**

4. **Edit the setting of the `registry.install.DatabaseUserID` property.**

   For embedded mode, this value can be either `APP` or empty:

   ```
   registry.install.DatabaseUserID=APP

   registry.install.DatabaseUserID=
   ```

   For Network Server mode, this value must be `APP`.

5. **Edit the setting of the `registry.install.DatabasePassword` property.**

   The password must contain at least 6 characters. The default value is `app123`.

6. **Save and close the file.**

7. **Stop and restart the Application Server domain for the Registry. To do so, execute the following command (all on one line):**

   Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file* `appserver.domain.bounce`

where *props-file* is the path name of the copy of `install.properties` file that you edited in "To Configure Service Registry as Root Using Custom Properties After a Configure Later Installation" on page 16 or "To Configure Service Registry as a Non-Root User Using Custom Properties After a Configure Later Installation" on page 17.

▼ To Perform an Offline Backup of the Database

If the database is running in embedded mode, you must perform an offline backup.

1. **Change to the Service Registry install directory:**

   `cd` *ServiceRegistry-base*`/install`

2. **Execute the following command (all on one line):**

   Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file* `appserver.domain.stop`

   where *props-file* is the path name of the copy of `install.properties` file that you edited in "To Configure Service Registry as Root Using Custom Properties After a Configure Later Installation" on page 16 or "To Configure Service Registry as a Non-Root User Using Custom Properties After a Configure Later Installation" on page 17.

3. **Execute the following command (all on one line):**

   Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file* `backup.db`

4. **Restart the domain by executing the following command (all on one line):**

   Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file* `appserver.domain.start`

▼ To Switch From Embedded Mode to Network Server Mode

To switch from embedded mode (the default) to Network Server mode, change the setting of the `registry.install.clientDatabase` property from `false` to `true`, and require database authentication.

After editing the property, recreate the database resources, then stop and restart the Application Server.

1. **Change to the Service Registry install directory:**

   `cd` *ServiceRegistry-base*`/install`

2. **Open your copy of the `install.properties` file in a text editor.**

3. **Change the setting of the property** `registry.install.clientDatabase` **from** `false` **to** `true`**.**

4. **Change the setting of the** `registry.install.RequireDatabaseAuthentication` **property from** `false` **to** `true`**.**

5. **Edit the setting of the** `registry.install.DatabaseUserID` **property, if necessary.**

   For Network Server mode, this value must be `APP`.

6. **Edit the setting of the** `registry.install.DatabasePassword` **property.**

   Any length is valid. The default value is `app123`.

7. **Save and close the file.**

8. **Recreate the database connection pool and its associated resource. To do so, execute the following command (all on one line):**

   Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file* `appserver.jdbcResource.update`

   where *props-file* is the path name of the copy of `install.properties` file that you edited in "To Configure Service Registry as Root Using Custom Properties After a Configure Later Installation" on page 16 or "To Configure Service Registry as a Non-Root User Using Custom Properties After a Configure Later Installation" on page 17.

9. **Stop and restart the Application Server domain for the Registry. To do so, execute the following command (all on one line):**

   Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file* `appserver.domain.bounce`

10. **Start the Java DB database. To do so, execute the following command:**

    `asadmin start-database --dbhome` *database-directory*

    By default, this command places the database and its log file in the current directory. Use the `--dbhome` option to specify the location of your database (normally, this is *RegistryDomain-base*`/3.0/data/registry/soar`).

Next Steps If the database is running in Network Server mode, you can perform an online backup of the database.

Later, if you want to return to embedded mode, follow the same steps, with the following exceptions:

- Change the setting of the `registry.install.clientDatabase` property from `true` to `false`.
- If you want to stop requiring database authentication, change the setting of the `registry.install.RequireDatabaseAuthentication` property from `true` to `false`.
- If you want to continue requiring database authentication, change the user ID and password if you wish. The `registry.install.DatabaseUserID` value must be either `APP` or empty. The `registry.install.DatabasePassword` value can be any length.
- Do not execute step 10. You do not need to start the database separately.

▼ To Perform an Online Backup of the Database

1. **Change to the Service Registry install directory:**

   `cd` *ServiceRegistry-base*`/install`

2. **Verify that the property `registry.install.clientDatabase` in your copy of the `install.properties` file is set to `true`.**

3. **Execute the following command (all on one line):**

   Ant-base`/ant -f build-install.xml Dinstall.properties=`*props-file* `backup.db`

   where *props-file* is the path name of the copy of `install.properties` file that you edited in "To Configure Service Registry as Root Using Custom Properties After a Configure Later Installation" on page 16 or "To Configure Service Registry as a Non-Root User Using Custom Properties After a Configure Later Installation" on page 17.

◆ ◆ ◆ CHAPTER 2

Using the Administration Tool

This chapter describes how to use the Administration Tool ("the Admin Tool") for the Service Registry.

This chapter contains the following sections:

About the Admin Tool

The Service Registry Administration Tool provides a simple command-line interface for common administration tasks, such as adding associations to the Registry and removing objects from the Registry.

The tool can operate in either of two modes:

- In batch mode, you specify one or more commands on the tool's command line.
- In interactive mode, you enter commands in the tool's interactive shell.

Several commands, such as `ls` and `rm`, mimic both the name and the behavior of well-known UNIX® commands that operate on files and folders. Other commands have no corresponding UNIX equivalent.

Starting the Admin Tool

To start the Admin Tool, you execute the `admin-tool.jar` file:

`java` *java-options* **`-jar`** *ServiceRegistry-base*`/lib/admin-tool.jar [`*admin-tool-options*`]...`

The `java` command is normally in the directory `/usr/jdk/entsys-j2se/bin`.

The *ServiceRegistry-base* location is `/opt/SUNWsrvc-registry` on Solaris OS and `/opt/sun/srvc-registry` on Linux and HP-UX systems.

You can safely ignore the warnings that appear when you start the tool.

To exit the Admin Tool, use the `quit` command.

Batch Mode

To run the Admin Tool in batch mode, specify the `-command` option on the command line when you start the Admin Tool.

For example, the following command executes the `ls` command:

`java -jar` *ServiceRegistry-base*`/lib/admin-tool.jar -command "ls *.html"`

The Admin Tool echoes your commands and the tool's responses to the screen and then exits after your commands have been executed.

Make sure that you properly escape any characters that are significant to your shell.

Interactive Mode

To run the Admin Tool in interactive mode, start the Admin Tool shell by specifying any options other than `-command` (or no options) on the command line:

`java -jar` *ServiceRegistry-base*`/lib/admin-tool.jar`

The Admin Tool displays the following prompt and waits for your input:

```
admin>
```

Admin Tool Command-line Options

The Admin Tool recognizes the command-line options that are listed in "Synopsis" on page 43 and described in "Options" on page 43.

Synopsis

```
[-alias alias] [-command commands] [-debug] [-help] [-keypass keypass]
[-localdir localdir] [-locale locale] [-registry url] [-root locator [-create]]
[-sqlselect SQL-statement] [-verbose]
```

Options

`-alias` The alias to use when accessing the user's certificate in the keystore. Specify the alias that you used when you registered as a user. This option is required if you will use the Admin Tool to publish data to the Registry.

`-command` The Admin Tool command sequence to run instead of getting commands from the interactive shell. Use a semicolon (`;`) to separate multiple commands. You do not have to include a `quit` command in *commands*. If you need to use a semicolon that is not a command separator, precede the semicolon by a backslash:

```
\;
```

The shell in which you run the Admin Tool might require you to escape the backslash with a second backslash:

```
\\;
```

If any command contains spaces, enclose the entire command sequence in single or double quotes so that the tool will treat the sequence as one command-line parameter instead of several. If your shell also interprets a semicolon as separating shell commands, you always have to put sequences of multiple Admin Tool commands in quotation marks.

`-create` If necessary, create the `RegistryPackage` specified by the `-root` option as well as any parent `RegistryPackage` objects as needed. This option is valid only if the user who is running the Admin Tool is authorized to create objects.

`-debug` Outputs extra information that is useful when debugging.

`-help` Provides a list of these options.

`-keypass` The password to use when accessing a user's certificate in the keystore. Specify the password that you used when you registered as a user. This option is required if you will use the Admin Tool to publish data to the Registry.

`-localdir` The base directory in the local file system for commands that relate to files in the local file system.

`-locale`	The locale (for example, `en` or `fr`) to use for selecting the resource bundle to use for error and status messages. The default is determined by the Java Virtual Machine (JVM).
`-registry`	The URL of the ebXML registry to which to connect. The default is `http://localhost:6480/soar/registry/soap`.
`-root`	The locator (for example, `/registry/userData`) of the `RegistryPackage` to use as the base for those commands that treat the repository as a tree of `RegistryPackage` objects that each contain other `RegistryObject` and `RegistryPackage` objects. The default is the `RegistryPackage` that is defined for all users' data: `/registry/userData`.
`-sqlselect`	Execute *SQL-statement* to select registry objects. The statement should be a complete SQL statement that starts with `select`. The SQL statement must be enclosed in quotation marks, but it does not have to be terminated by a semicolon. If you specify this option and then use the `select` command with no argument, the command will execute *SQL-statement* until you use the `select` command with an argument other than *SQL-statement*.
`-v` | `-verbose`	Specifies the verbose output of status messages.

Note – The output of the `-help` option lists two options that are not supported in this release: `-class` and `-property`.

Using the Admin Tool to Publish Content to the Registry

Some Admin Tool commands allow you to publish content to the Registry: `cp` and `import`, for example. In addition, the `rm` command allows you to delete content from the Registry. Before you can use these commands, you must perform some additional steps.

▼ To Enable Yourself to Publish Content to the Registry

1. **Perform user registration as described in "Creating a User Account" in** *Service Registry 3.1 User's Guide***.**

   Remember the location of the PKCS12 certificate you downloaded, as well as the user name and password you specified.

2. **Start the Admin Tool:**

   `java -jar` *ServiceRegistry-base***`/lib/admin-tool.jar`**

3 **Execute the `keystoreMover` command to export your PKCS12 certificate to a JKS keystore. See "`keystoreMover`" on page 61 for details.**

Typically, you need to specify only the four options shown in the command example.

4 **Stop the Admin Tool:**

```
quit
```

5 **Start the Admin Tool again. This time, specify options as follows:**

```
java -Djaxr-ebxml.security.storetype=JKS \
-Djaxr-ebxml.security.keystore=security/filename \
-Djaxr-ebxml.security.storepass=ebxmlrr \
-jar ServiceRegistry-base/lib/admin-tool.jar -alias alias -keypass password
```

Here, *filename* is the name of your certificate file, which is normally `keystore.jks`. The location `security/`*filename* is relative to the directory `$HOME/soar/3.0/jaxr-ebxml`. The *alias* and *password* values are the ones you specified when you created a user account.

To save typing, create a script to execute this command.

Admin Tool Features

This section describes the following features of the Admin Tool:

Permissions

When you use the Admin Tool, you can perform only those actions that are allowed for the user whose key alias and password you specified when you started the tool. Only a user with the role of administrator can perform commands that make changes to objects that the user does not own. See "Creating an Administrator" on page 25 for details.

Displaying Exceptions

The Admin Tool enables you to avoid viewing long stack traces when a command fails.

When a command fails, the Admin Tool prints the first line of the stack trace and the following message:

```
An error occurred when executing the function.  Use the show exception
command to view messages.
```

If you need more information, execute the `show exception` command next to see the full stack trace.

The `show exception` command always displays the stack trace of the immediately preceding command.

Identifying Registry Objects

The primary way to identify registry objects is by name. However, you normally identify `RegistryPackage` objects by the path from the `registry` root to the `RegistryPackage`. For example, `/registry/userData` is the path to the `userData RegistryPackage`.

Some matches for names support wildcards. Use a question mark (`?`) to match a single character. Use an asterisk (`*`) to match zero or more characters.

Some commands (for example, `cd` and `chown`) support identifying objects by their Uniform Resource Name (URN), which must include a leading `urn:`. For example, `urn:uuid:2702f889-3ced-4d49-82d1-e4cd846cb9e4` is a valid URN.

The `chown` and `cp` commands also support the use of `%`*number* to refer to a `User` listed by a previous `users` command.

For some commands, you can enter names that contain spaces by enclosing the entire name in double quotes or by preceding each space in the name by a backslash.

The `select` command supports the use of SQL wildcards: the percent sign (`%`) to match multiple characters, and the underscore (`_`) to match a single character.

The Effect of Locale on Specifying Names

A `RegistryObject` (or a `RegistryPackage`) can have multiple names, each of which is associated with a different locale.

The paths and object names that you specify are evaluated with respect to the current locale only. When you attempt to select by name a registry object that has multiple names, the Registry

attempts to match the name that you provide against only one alternative for the registry object's name (the choice whose locale most closely matches the current locale), not against all the multiple names for the registry object.

For example, suppose the current `RegistryPackage` has a member object that has two names, each associated with a different locale: `red` in the `en` (English) locale and `rouge` in the `fr` (French) locale. When the current locale is `en`, the command `ls rouge` does not display that member object, but when the locale is `fr` (or one of its variants), it does.

Case Sensitivity

Command names and literal parameters that are recognized by the Admin Tool are not case sensitive. For example, `ls`, `Ls`, and `LS` are equivalent.

Options to which you provide the value are passed literally to the code that uses the option.

Using Admin Tool Commands

The following sections describe the available commands.

For each command, the synopsis and the descriptions of the options and operands observe the following typographical conventions:

- *Italics* indicate an option argument or operand that should be replaced by an actual value when you run the command.
- Curly braces (`{ }`) delimit a choice of options or operands where you must include one of the options or operands. The options or operands are separated by a vertical bar (`|`).
- Square brackets (`[ ]`) delimit an option or operand, or a choice of options or operands, that may be omitted.
- An ellipsis (`. . .`) after an option or operand indicates that you may repeat the argument or operand.

Anything else is literal text that you must include when running the command.

add association

Adds an `Association` object to the Registry.

Synopsis

`add association` `-type` *association-type sourceURN targetURN*

Description

The `add association` command adds an `Association` object of the specified type to the Registry.

You can use any of the following types:

- `AccessControlPolicyFor`
- `AffiliatedWith` (which has the subconcepts `EmployeeOf` and `MemberOf`)
- `Contains`
- `ContentManagementServiceFor`
- `EquivalentTo`
- `Extends`
- `ExternallyLinks`
- `HasFederationMember`
- `HasMember`
- `Implements`
- `InstanceOf`

- `InvocationControlFileFor` (which has the subconcepts `CatalogingControlFileFor` and `ValidationControlFileFor`)
- `OffersService`
- `OwnerOf`
- `RelatedTo`
- `Replaces`
- `ResponsibleFor`
- `SubmitterOf`
- `Supersedes`
- `Uses`

Options

`-type` The type of the `Association` object.

Operands

sourceURN The URN of the source object.

targetURN The URN of the target object.

Example

The following command (all on one line) creates a `RelatedTo` relationship between the objects with the two specified URNs.

```
admin> add association -type RelatedTo
urn:uuid:ab80d8f7-3bea-4467-ad26-d04a40045446
urn:uuid:7a54bbca-2131-4a49-8ecc-e7b4ac86c4fd
```

add user

Adds a user to the Registry.

Synopsis

add user [-edit] [-load *pathname*] [-firstname *string*] [-lastname *string*] [-middleName *string*] -alias *string* -keypass *string* [-post1.type *string*] [-post1.city *string*] [-post1.country *string*] [-post1.postalcode *string*] [-post1.stateOrProvince *string*] [-post1.street *string*] [-post2.streetNumber *string*] [-post2.type *string*] [-post2.city *string*] [-post2.country *string*]

[-post2.postalcode *string*] [-post2.stateOrProvince *string*] [-post2.street *string*]
[-post2.streetNumber *string*] [-post3.type *string*] [-post3.city *string*]
[-post3.country *string*] [-post3.postalcode *string*] [-post3.stateOrProvince *string*]
[-post3.street *string*] [-post3.streetNumber *string*] [-telephone1.type *string*]
[-telephone1.areaCode *string*] [-telephone1.countryCode *string*]
[-telephone1.extension *string*] [-telephone1.number *string*] [-telephone1.URL *string*]
[-telephone2.type *string*] [-telephone2.areaCode *string*]
[-telephone2.countryCode *string*] [-telephone2.extension *string*]
[-telephone2.number *string*] [-telephone2.URL *string*] [-telephone3.type *string*]
[-telephone3.areaCode *string*] [-telephone3.countryCode *string*]
[-telephone3.extension *string*] [-telephone3.number *string*] [-telephone3.URL *string*]
[-email1.type *string*] [-email1.address *string*] [-email2.type *string*]
[-email12address *string*] [-email3.type *string*] [-email3.address *string*]

Description

The `add user` command adds a `User` object. A `User` object normally contains at least one `PostalAddress`, `TelephoneNumber`, and `EmailAddress` object.

Specify the information about the user either on the command line itself or by using the `-load` option to specify a Java property file with the information. The information options and the `-load` option are evaluated in the order in which they appear on the command line. For example, you can specify some properties on the command line, load others from a property file, and then override information in the property file with subsequent command-line options.

You can specify up to three addresses, telephone numbers, and email addresses for a new user. If you need more, you can add them later using the Web Console or JAXR.

When you specify an address, telephone number, or email address, you must provide a value for its type: for example, `-emailType OfficeEmail`.

You can use shorthand options (such as `-fn`) on the command line for some of the common information that is required for every user. However, you must use the longer form when you provide the information in a property file. For example, you can specify the user's first email address on the command line using either `-email1.address`, `-emailAddress`, or `-email`. However, when you specify the first email address in a property file, you must use `email1.address=`. Because there is only one option for the user's second email address, you must use `-email2.address` on the command line and `email2.address=` in a property file.

If you specify the `-edit` option, the Admin Tool launches an editor so that you can edit the new user's information. See the option description for details.

The command creates a certificate keystore for the new user in your home directory, in the file `$HOME/soar/3.0/jaxr-ebxml/security/keystore.jks`. If you are running the tool as `root`, the home directory is either `/` or `/root`.

Note – The property files that you load with `-load` or edit with `-edit` use the IS0-8859-1 charset, as do all Java property files. See the documentation for `java.util.Properties.load(InputStream)` for details on how to represent characters not in ISO-8859-1 in property files.

Options

`-edit`
Causes the Admin Tool to launch an editor so that you can edit the new user's information. The tool launches the editor after evaluating the other command-line parameters. Therefore, editing starts with the result of evaluating any information that was specified on the command line or in a property file. The editing program must terminate without error before the command can continue. The Admin Tool launches the editor specified by the `set editor` command (see "`set`" on page 67); by default, this is the `vi` editor.

Note – At this release, `-edit` works with `emacsclient` and the NetBeans™ command `bin/runide.sh --open` (but not very well), and has not been shown to work with `vi`.

`-load`
Specifies a Java property file whose contents specify properties for the user. The property names are the same as the long forms of the `add user` command options (for example, `lastName` and `post1.type`).

`-fn` | `-firstName`
Specifies the first name of a user.

`-ln` | `-lastName`
Specifies the last name (surname) of a user. The last name, which is required, must be specified either on the command line or in a property file.

`-mn` | `-middleName`
Specifies the middle name of a user.

`-alias`
The alias to use when accessing the user's certificate in the keystore. This option is required. The alias must be at least three characters long.

`-keypass`
The password to use when accessing a user's certificate in the keystore. This option is required. The password must be at least six characters long.

`-postalType` | `-post1.type`
The type of the first `PostalAddress`. The type, which is required, must be specified either on the command line or in a property file. The value is an arbitrary string (for example, `Office` or `Home`).

`-city` | `-post1.city`
The city of the first `PostalAddress`.

`-country` | `-post1.country`
The country of the first `PostalAddress`.

`-postalCode` | `-postcode` | `-zip` | `-post1.postalcode`
The postal code of the first `PostalAddress`.

`-stateOrProvince` | `-state` | `-province` | `-post1.stateOrProvince`
The state or province of the first `PostalAddress`.

`-street` | `-post1.street`
The street name of the first `PostalAddress`.

`-streetNumber` | `-number` | `--post1.streetNumber`
The street number of the first `PostalAddress`.

`-post2.type`
The type of the second `PostalAddress`. If a second `PostalAddress` is specified, the type, which is required, must be specified either on the command line or in a property file. The value is an arbitrary string (for example, `Office` or `Home`).

`-post2.city`
The city of the second `PostalAddress`.

`-post2.country`
The country of the second `PostalAddress`.

`-post2.postalcode`
The postal code of the second `PostalAddress`.

`-post2.stateOrProvince`
The state or province of the second `PostalAddress`.

`-post2.street`
The street name of the second `PostalAddress`.

`-post2.streetNumber`
The street number of the second `PostalAddress`.

`-post3.type`
The type of the third `PostalAddress`. If a third `PostalAddress` is specified, the type, which is required, must be specified either on the command line or in a property file. The value is an arbitrary string (for example, `Office` or `Home`).

`-post3.city`
The city of the third `PostalAddress`.

`-post3.country`
The country of the third `PostalAddress`.

`-post3.postalcode`
The postal code of the third `PostalAddress`.

`-post3.stateOrProvince`
The state or province of the third `PostalAddress`.

`-post3.street`
The street name of the third `PostalAddress`.

`-post3.streetNumber`
The street number of the third `PostalAddress`.

`-phoneType` | `-telephone1.type`
The type of the first `TelephoneNumber`. The type, which is required, must be specified either on the command line or in a property file. The value is an arbitrary string, but you can specify one of the following known types: `Beeper`, `FAX`, `HomePhone`, `MobilePhone`, or `OfficePhone`.

`-areaCode` | `-telephone1.areaCode`
The area code of the first `TelephoneNumber`.

`-countryCode` | `-telephone1.countryCode`
The country code of the first `TelephoneNumber`.

`-extension` | `-telephone1.extension`
The extension of the first `TelephoneNumber`.

`-number` | `-telephone1.number`
The telephone number suffix, not including the country or area code, of the first `TelephoneNumber`. The number, which is required, must be specified either on the command line or in a property file.

`-URL` | `-telephone1.URL`
The URL of the first `TelephoneNumber` (the URL that can dial this number electronically).

`-telephone2.type`
The type of the second `TelephoneNumber`. If a second `TelephoneNumber` is specified, the type, which is required, must be specified either on the command line or in a property file. The value is an arbitrary string, but you can specify one of the following known types: `Beeper`, `FAX`, `HomePhone`, `MobilePhone`, or `OfficePhone`.

`-telephone2.areaCode`
The area code of the second `TelephoneNumber`.

`-telephone2.countryCode`
The country code of the second `TelephoneNumber`.

`-telephone2.extension`
The extension of the second `TelephoneNumber`.

`-telephone2.number`
The telephone number suffix, not including the country or area code, of the second `TelephoneNumber`. If a second `TelephoneNumber` is specified, the number, which is required, must be specified either on the command line or in a property file.

`-telephone2.URL`
The URL of the second `TelephoneNumber` (the URL that can dial this number electronically).

`-telephone3.type`
The type of the third `TelephoneNumber`. If a third `TelephoneNumber` is specified, the type, which is required, must be specified either on the command line or in a property file. The value is an arbitrary string, but you can specify one of the following known types: `Beeper`, `FAX`, `HomePhone`, `MobilePhone`, or `OfficePhone`.

`-telephone3.areaCode`
The area code of the third `TelephoneNumber`.

`-telephone3.countryCode`
The country code of the third `TelephoneNumber`.

`-telephone3.extension`
The extension of the third `TelephoneNumber`.

`-telephone3.number`
The telephone number suffix, not including the country or area code, of the third `TelephoneNumber`. If a third `TelephoneNumber` is specified, the number,which is required, must be specified either on the command line or in a property file.

`-telephone3.URL`
The URL of the third `TelephoneNumber` (the URL that can dial this number electronically).

`-emailType` | `-email1.type`
The type of the first `EmailAddress`. The type, which is required, must be specified either on the command line or in a property file. The value is an arbitrary string, but you can specify one of the following known types: `HomeEmail` or `OfficeEmail`.

`-emailAddress` | `-email` | `-email1.address`
The first email address. The first email address is required.

`-email2.type`
The type of the second `EmailAddress`. If a second `EmailAddress` is specified, the type, which is required, must be specified either on the command line or in a property file. The value is an arbitrary string, but you can specify one of the following known types: `HomeEmail` or `OfficeEmail`.

`-email2.address`
The second email address.

`-email3.type`
The type of the third `EmailAddress`. If a third `EmailAddress` is specified, the type, which is required, must be specified either on the command line or in a property file. The value is an arbitrary string, but you can specify one of the following known types: `HomeEmail` or `OfficeEmail`.

`-email3.address`
The third email address.

Examples

The following command loads the `User` properties from the file `JaneSmith.properties` in the user's home directory.

```
admin> add user -load ~/JaneSmith.properties
```

The following command (all on one line) specifies the minimum properties that are required to create a `User` object.

```
admin> add user -ln Smith -postaltype Office -country US
-phonetype Office -number 333-3333 -emailtype OfficeEmail
-emailaddress JaneSmith@JaneSmith.com -alias 123 -keypass 123456
```

cd

Changes the `RegistryPackage` location.

Synopsis

cd {*locator* | *URN*}

Description

The `cd` command changes directory (metaphorically) to the `RegistryPackage` at the specified path or with the specified URN.

The command changes to a specified URN when multiple `RegistryPackage` objects exist with the same path (for the current locale).

Operands

locator The path of names of registry objects from the root of the repository to an object in the repository, with each name preceded by a forward slash (`/`).

For example, the locator for the `userData` `RegistryPackage` that is a member of the `registry` `RegistryPackage` (which is not itself a member of any `RegistryPackage`) is `/registry/userData`. The locator for the `folder1` `RegistryPackage` that is a member of the `userData` `RegistryPackage` is `/registry/userData/folder1`.

URN The URN of the `RegistryPackage`, which must be a URN starting with `urn:`.

Examples

The following command changes the directory to the `RegistryPackage` with the URN `urn:uuid:92d3fd01-a929-4eba-a5b4-a3f036733017`.

```
admin> cd urn:uuid:92d3fd01-a929-4eba-a5b4-a3f036733017
```

The following command changes the directory to the location `/registry/userData/myData`.

```
admin> cd /registry/userData/myData
```

chown

Changes the owner of a `RegistryObject`.

Synopsis

`chown` {*URN* | *%index*}

Description

The `chown` command changes the ownership of the objects selected with a preceding `select` command to the user specified either by the URN or by the reference to the user's URN that was listed by a preceding `users` command.

Only a user with the role of administrator can execute this command successfully.

Operands

URN The `User` object specified by the URN.

%index A numerical reference to a URN for a `User` object listed in a preceding `users` command.

Examples

The following command changes the ownership of the selected objects to the user specified by the URN `urn:uuid:26aa17e6-d669-4775-bfe8-a3a484d3e079`.

```
admin> chown urn:uuid:26aa17e6-d669-4775-bfe8-a3a484d3e079
```

The following command changes the ownership of the selected objects to the user with the number 2 in a preceding `users` command.

```
admin> chown %2
```

See Also

- "`select`" on page 67
- "`users`" on page 69

cp

Copies files and folders into the Registry.

Synopsis

`cp` `[-owner {`*URN* `|` `%`*index*`}] [-exclude` *pattern*`]… [-include` *pattern*`]…` *pattern*…

Description

The `cp` command copies folders and files into the Registry as `RegistryPackage` and `ExtrinsicObject` objects, respectively.

The local directory on the local file system from which to copy files and folders defaults to the current directory from which you started the Admin Tool. You can use the `-localdir` option to change the local directory when you start the Admin Tool. You can use the `lcd` command to change the local directory after the Admin Tool has started. You can get the absolute path of the current local directory by using the `show localdir` command.

The command is recursive. That is, if you specify a directory, the command copies all the files and folders under the directory.

Options

`-owner` Sets the owner of the copied registry objects to the user specified by the *URN* or `%`*index* argument. See the description of the `chown` command for a description of these arguments. You must have the role of administrator to specify an owner other than yourself.

`-exclude` Copies all files except those whose names contain the specified pattern, where *pattern* is a pattern comprising literal characters and the special characters

asterisk (*) (representing zero or more characters) and question mark (?) (representing one and only one character).

You can specify this option more than once.

`-include` Copies all files whose names contain the specified pattern, where *pattern* is a pattern comprising literal characters and the special characters asterisk (*) (representing zero or more characters) and question mark (?) (representing one and only one character).

You can specify this option more than once.

Operands

pattern The files or folders to be copied, specified by a pattern comprising literal characters and the special characters asterisk (*) (representing zero or more characters) and question mark (?) (representing one and only one character). You can specify more than one *pattern*.

Examples

The following command copies the directory `mydir` to the Registry, to be owned by the user with the number 4 in a preceding `users` command.

```
admin> cp -owner %4 mydir
```

The following command copies the directory `mydir` to the Registry, excluding files and directories that end with the string `.z` or `.c`.

```
admin> cp mydir -exclude \\.z -exclude \\.c
```

See Also

- "`lcd`" on page 63
- "`show`" on page 68

echo

Echoes a string.

Synopsis

`echo` *string*

Description

The `echo` command echoes the specified *string* to the output. This command is most useful when you specify it in the `-command` option when you run the Admin Tool in batch mode.

Operand

string A sequence of characters.

Example

The following command prints the date and the result of the `ls` command into a log file.

```
java -jar admin-tool.jar -command "echo ”date”; ls" > admin.log
```

help

Displays information about commands.

Synopsis

help [*command_name*]

Description

The `help` command displays information about the available commands or a specified command.

For commands with subcommands, such as `add` and `show`, the `help` command displays information about the subcommands.

If you do not specify an argument, the `help` command displays usage information for all commands.

Operand

command_name The name of an Admin Tool command.

Examples

The following command displays usage information for all commands.

```
admin> help
```

The following command displays usage information for the `lcd` command.

```
admin> help lcd
```

The following command displays usage information for the `add` subcommands.

```
admin> help add
```

import

Imports an XML file that defines registry objects.

Synopsis

`import` `[ {-a | --attach}` *`pathname, mimeType, id`*`]…` *`submitObjectsRequest`*

Description

The `import` command adds one or more new objects or repository items to the Registry by submitting an XML file that conforms to the `SubmitObjectsRequest` protocol as described in the *ebXML Registry Services and Protocols Version 3.0* specification.

The XML file and any attachments provided form the content of a SOAP message sent to the Registry and handled there. This operation is thus very low-level and is provided for those most familiar with the ebXML specifications.

Option

`-a|--attach` Attaches a file as the repository item for an extrinsic object. The *pathname* is the path name of the file to be added. The *mimeType* specifies the MIME type of the file. The *id* is the unique identifier of the extrinsic object for which this is the repository item (??). You can specify this option more than once.

Operand

submitObjectsRequest An XML file that contains definitions of registry objects.

Examples

The following command imports a group of objects defined in the file `MyRequest.xml`:

```
admin> import MyRequest.xml
```

The following command (all on one line) imports an extrinsic object and its repository item, an image file:

```
admin> import --attach chicken.jpg, image/jpeg, urn:bird:poultry:chicken
ChickenRequest.xml
```

keystoreMover

Exports one or more keys from one keystore format to another.

Synopsis

keystoreMover [-sourceKeystoreType {JKS | PKCS12}] -sourceKeystorePath *pathname* -sourceKeystorePassword *password*
[-sourceAlias *alias* [-sourceKeyPassword *password*]] [-destinationKeystoreType {JKS | PKCS12}] -destinationKeystorePath *pathname*
-destinationKeystorePassword *password* [-destinationAlias *newAlias*]
[-destinationKeyPassword *password*]

Description

The developer interface to the Registry requires the use of a JKS keystore, while the Web Console requires a PKCS12 or DER certificate that you can import into a web browser.

If you created a user account using a registry-generated PKCS12 certificate and you wish to use the Admin Tool to publish content to the Registry, use the `keystoreMover`command to export the certificate to a JKS keystore. You can also use this command in order to run developer applications against the Registry.

If you created a user with the `add user` command and you want that user to be able to use the Web Console, you can use this command to export the JKS keystore created by `add user` to the PKCS12 format.

See "Creating a User Account" in *Service Registry 3.1 User's Guide* for details on using the Web Console to create a user account. See "Using the Admin Tool to Publish Content to the Registry" on page 44 for information on using the `keystoreMover` in conjunction with the Admin Tool. See "`add user`" on page 49 for information on using the `add user` command.

See *Service Registry 3.1 Developer's Guide* for information on developing applications for the Registry.

Options

`-sourceKeystoreType` — Specifies the type of the keystore to be exported. Argument must be either `PKCS12` or `JKS`. The default is `PKCS12`.

`-sourceKeystorePath`	Specifies the path name of the file that contains the source keystore. This option is required. Normally, this is the path name of the certificate file created when you created a user.
`-sourceKeystorePassword`	Specifies the password for the source keystore. Normally, this is the password you specified when you created a user. This option is required.
`-sourceAlias`	Specifies the alias to be exported. If you do not specify this option, the command exports all aliases in the keystore. The keystore downloaded from the Web Console contains only one alias.
`-sourceKeyPassword`	Specifies the password specific to the alias (as opposed to the keystore password). If you do not specify this option, the password is the same as the keystore password (this is the usual case).
`-destinationKeystoreType`	Specifies the type of the destination keystore. Argument may be either `JKS` or `PKCS12`. The default is `JKS`.
`-destinationKeystorePath`	Specifies the path name of the file that will contain the destination keystore. This option is required. Normally, this argument is *HOME*`/soar/3.0/jaxr-ebxml/security/keystore.jks`, where *HOME* is the user's home directory.
`-destinationKeystorePassword`	Specifies the password for the destination keystore. This argument is required. The default value of this property is `ebxmlrr`.
`-destinationAlias`	Specifies the new alias name, if you want to rename the alias. If you do not specify this option, the new alias has the same name as the alias of the source certificate.
`-destinationKeyPassword`	Specifies the password specific to the alias (as opposed to the keystore password). If you do not specify this option, the password is the same as the keystore password (this is the usual case).

Note – All passwords must be at least 6 characters in length.

Example

The following command exports the certificate in `generated-key.p12` in a user's home directory to a JKS keystore in `soar/3.0/jaxr-ebxml/security/keystore.jks`, also in the

user's home directory. The source keystore password is the one provided when the user registered with the Registry. The destination keystore password is the default value of `ebxmlrr`. Specify the command all on one line.

```
admin> keystoreMover -sourceKeystorePath /home/myname/generated-key.p12
-sourceKeystorePassword mypass -destinationKeystorePath
/home/myname/soar/3.0/jaxr-ebxml/security/keystore.jks
-destinationKeystorePassword ebxmlrr
```

lcd

Changes the current directory on the local file system.

Synopsis

lcd [*pathname*]

Description

The `lcd` command changes the current local directory on the local file system.

If you do not specify an argument, the `lcd` command changes the current directory to your default home directory.

Operand

pathname A directory name, which can be absolute or relative.

Examples

The following command changes the current local directory to the `/usr/share` directory.

```
admin> lcd /usr/share
```

The following command changes the current local directory to your default home directory on the local file system.

```
admin> lcd
```

ls

Lists the objects in the current `RegistryPackage`.

Synopsis

ls [{*pattern* | *URN*}…]

Description

With no arguments, the `ls` command lists the objects in the current `RegistryPackage`. When a *pattern* or *URN* is provided, the command lists the objects in the current `RegistryPackage` whose names (in the current locale) or unique identifiers match *pattern* or *URN*.

Operands

pattern A pattern comprising literal characters and the special characters asterisk (`*`) (representing zero or more characters) and question mark (`?`) (representing one and only one character). You can specify more than one *pattern*.

URN A URN starting with `urn:`, for example, `urn:uuid:4a6741e7-4be1-4cfb-960a-e5520356c4fd`. You can specify more than one *URN*. The URN must be the unique identifier of the object, not the logical identifier.

Examples

The following command lists all the objects in the current `RegistryPackage`.

```
admin> ls
```

The following command lists all the objects whose name matches the pattern `urn:bird:poultry:chicken` or whose ID is `urn:bird:poultry:chicken`.

```
admin> ls urn:bird:poultry:chicken
```

The following command lists all the objects whose name matches the pattern `*bird*`. (It would also list the objects whose ID is `*bird*`, if `*bird*` were a valid ID.)

```
admin> ls *bird*
```

The following command lists all the objects whose name matches the pattern `*bird*` or whose name matches the pattern `urn:bird:poultry:chicken` or whose ID is `urn:bird:poultry:chicken`.

```
admin> ls *bird* urn:bird:poultry:chicken
```

pwd

Displays the path to the current `RegistryPackage`.

Synopsis

pwd

Description

The `pwd` command displays the path (or paths) to the current `RegistryPackage` using the best-matching names for the current locale. The command also displays the locale for the path.

Example

```
admin> pwd
(en_US) /registry/userData
```

quit

Exits the Admin Tool.

Synopsis

quit

Description

The `quit` command exits the Admin Tool.

Example

```
admin> quit
```

rm

Removes objects from a `RegistryPackage`.

Synopsis

rm `[-d] [-r]` {*pattern* | *URN*}**…**

Description

The `rm` command removes the member objects of the current `RegistryPackage` whose names (in the current locale) match the patterns specified by a *pattern* or *URN*.

When a matching `RegistryObject` is a member of multiple `RegistryPackage` objects, this command removes only the association between the current `RegistryPackage` and the object. The object is removed from the Registry only when the removal of the association leaves the object with no association with any other `RegistryObject`, including other containing `RegistryPackage` objects.

When a matching member object is itself a `RegistryPackage` that contains other objects, neither the object nor the association between the current `RegistryPackage` and the member `RegistryPackage` is removed unless either the `-r` or the `-d` option is specified.

When both the `-d` and `-r` options are specified, the `-d` option is applied recursively, so all objects that would be selected by `-r` (and their associations) are removed whether or not they have other associations.

Options

`-d` Removes the association between the current `RegistryPackage` and the specified `RegistryPackage`. Removes the specified `RegistryPackage` only if its only remaining associations are to its member objects. Member objects of the now-removed `RegistryPackage` that are not anchored by being the target of other `HasMember` associations are now accessible as members of the root of the Registry.

`-r` Removes the specified `RegistryPackage` object and all its descendant objects (except when an object has other associations).

Operands

pattern A pattern comprising literal characters and the special characters asterisk (`*`) (representing zero or more characters) and question mark (`?`) (representing one and only one character). You can specify more than one *pattern*.

URN A URN starting with `urn:`, for example, `urn:uuid:4a6741e7-4be1-4cfb-960a-e5520356c4fd`. You can specify more than one *URN*.

Examples

The following command removes all `RegistryPackage` objects that contain the string `"stat"` and all their descendants.

```
admin> rm -r *stat*
```

select

Executes an SQL `select` statement.

Synopsis

`select` [*SQL*]

Description

The `select` command selects and lists the objects that are specified by evaluating the entire command as an SQL query. If no argument is specified, the command lists any objects selected by a preceding `select` command or by the `-sqlselect` option.

Operand

SQL An SQL `select` statement (without the leading `select` because that is already present as the name of the command).

Examples

The following command lists all `ClassificationScheme` objects in the Registry:

```
admin> select s.* from ClassificationScheme s
```

set

Sets a property value.

Synopsis

`set` *property value*

Description

The `set` command sets the value of a property of the Admin Tool shell.

The tool supports the following properties and values.

`set debug` {`true` | `on` | `yes` | `false` | `off` | `no`}

Enables or disables output of debugging messages.

`set editor` *string*

Sets the command to use when the Admin Tool launches an interactive editor. The default value is `/bin/vi` on UNIX and Linux systems.

`set verbose` `{true | on | yes | false | off | no}`

Enables or disables output of more verbose messages when executing commands.

Operands

property One of the following properties: `debug`, `editor`, `verbose`.

value A supported value of the specified property. See the Description section for details.

Examples

The following command sets the editor to `/usr/bin/vi` instead of the default `/bin/vi`.

```
admin> set editor /usr/bin/vi
```

The following command turns on debugging.

```
admin> set debug true
```

The following command turns off verbose output.

```
admin> set verbose off
```

show

Displays a property value.

Synopsis

`show` [*property*]

Description

The `show` command displays the value of a property of the Admin Tool shell.

If no argument is specified, the command displays the values of all properties.

The command supports the following properties:

`debug` Whether or not debugging output is enabled.

`editor` The editor to use when the Admin Tool launches an interactive editor.

`exception`	The exception stack trace, if any, from the immediately preceding executed command.
`localdir`	The current directory on the local file system. Use the `lcd` command to set this property. See "`lcd`" on page 63 for details.
`locale`	The current locale.
`verbose`	Whether or not verbose output is enabled.

Operands

property	The property whose current value is to be displayed. The properties `exception` and `locale` can be displayed, but you cannot use the `set` command to set them.

Example

The following command displays the exceptions from the previous command.

```
admin> show exception
```

users

Lists the current `User` objects.

Synopsis

```
users
```

Description

The `users` command lists the `User` objects currently in the Registry.

The output has the following format:

`%`*index*: *URN lastname*, *firstname middlename*

In the output, the *index* is a numeric value that you can use, including the percent sign (`%`), to refer to a user when you run the `chown` or `cp` command. The *lastname*, *firstname*, and *middlename* are the last, first, and middle names of the user.

Example

The following command displays the current users:

```
admin> users
%0:  urn:freebxml:registry:predefinedusers:registryoperator  Operator, Registry
%1:  urn:freebxml:registry:predefinedusers:registryguest  Guest, Registry
%2:  urn:freebxml:registry:predefinedusers:farrukh  Najmi, Farrukh Salahudin
%3:  urn:freebxml:registry:predefinedusers:nikola  Stojanovic, Nikola
%4:  urn:uuid:799cc524-b7cd-4e51-8b34-d93b79ac52de  User, Test
%5:  urn:uuid:85428d8e-1bd5-473b-a8c8-b9d595f82728  Parker, Miles
```

See Also

- "chown" on page 56
- "cp" on page 57

Index

D

E

F

G

H

I

J

K

L

W

X