

## Chapter 11

# Installing Web Applications

This chapter describes how to install Web applications that are included with the SDX software. The chapter contains the following sections:

- Installing Web Applications on page 137
- Removing Web Applications on page 139
- Session Timeouts for Web Applications on page 139
- Access Controls on page 140

## Installing Web Applications

---

We supply one Web archive (WAR) file for each Web application in the SDX software distribution and the application library CD. You must deploy Web applications in a Web application server.

The exact way you install Web applications depends on the Web application server you are using and the particular Web application. The following procedure provides general steps for installing a Web application:

1. Install the Web application server on the host.
2. Start the Web application server.
3. If the Web application requires configuration of a properties file, complete the following procedure:
  - a. Copy the WAR file from the SDX software distribution or application library CD to a temporary folder on the host.
  - b. Unpack the WAR file.

For information about unpacking and packing WAR files, see

<http://java.sun.com/j2se/1.4/docs/guide/jar/>

- c. Edit the properties file for the Web application.
- d. Repack the WAR file.

4. Deploy the WAR file by using the procedure appropriate for your Web application server.

For information about deploying WAR files, see the documentation for your Web application software.

### **Installing Web Applications Inside JBoss**

We provide the JBoss Web application server in the SDX software distribution. JBoss is an open-source Java application server that provides full support for J2EE application programming interfaces (APIs).

To deploy a Web application inside JBoss:

1. Install the *UMCjboss* package from the SDX software distribution.

See *Chapter 5, Installing the SDX-300 Software* for information about using the Solaris **pkgadd** utility to install the package.

2. During the installation, choose a JBoss configuration when prompted; typically choose the default configuration.
3. Start JBoss.

```
/etc/init.d/jboss start
```

You can view the log file to observe the process:

```
/opt/UMC/jboss/server/default/log/server.log
```

4. Customize the properties file for the Web application.

For instructions about configuring the property files for the SDX Web applications, see the documentation for that application.

5. Deploy the WAR file by copying it into the JBoss *default/deploy* directory.

```
cp <filename>.war /opt/UMC/jboss/server/default/deploy
```

JBoss automatically starts the Web application when a new WAR file is copied into the deploy directory.

### **Stopping JBoss**

To stop JBoss:

1. On the host on which JBoss is installed, log in as **root** or as an authorized nonroot admin user.
2. Stop JBoss.

```
/etc/init.d/jboss stop
```

## Removing Web Applications

---

The way you remove a Web application depends on the Web application server that you use.

To remove a deployed Web application from JBoss:

- Remove the WAR file from the JBoss *default/deploy* directory.

## Session Timeouts for Web Applications

---

For session-based Web applications, a session is started for the Web browser when it connects to a Web server application. A session provides a timeout feature that closes the session on the server when the maximum period of inactivity has passed. The default timeout for many application servers is 30 minutes.

This timeout is reset whenever there is activity on the Web browser, such as refreshing the current page or navigating through other pages under the application's control. Merely keeping a browser window open does not keep the session open, because it does not generate any activity on the browser.

When the session closes, any application-related state must be reestablished by the Web browser. Examples include such items as redoing the login, the parameters of the session, and connection to back-end systems such as directory servers or Common Object Request Broker Architecture (CORBA) servers.

You may be able to customize the session timeout, depending on the type of Web server or Web application server that you are using. See the documentation for your Web server or Web application server for information about configuring these settings.



**NOTE:** Long timeouts or no timeouts not only result in security concerns for the browser, but also result in more resource usage on the servers to keep stale sessions.

---

The session timeout in *web.xml* used in a J2EE application server might be set as follows:

```
<web-app>
...
  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>
...
</web-app>
```

## Access Controls

---

To enforce J2EE-style access controls, Web applications deployed in JBoss must contain a *WEB-INF/jboss-web.xml* file that defines a security domain as shown here:

```
<jboss-web>
<security-domain>java:/jaas/TEST_SECURITY_DOMAIN</security-domain>
</jboss-web>
```

For these Web applications, JBoss performs authentication as defined in the application's deployment descriptor, the *WEB-INF/web.xml* file. Here is the relevant sample portion of a *WEB-INF/web.xml* file:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>TEST_WEB_RESOURCE_NAME</web-resource-name>
    <!-- Define the context-relative URL(s) to be protected -->
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>TEST_ROLE_NAME</role-name>
  </auth-constraint>
</security-constraint>

<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>TEST_REALM_NAME</realm-name>
</login-config>
```

This *web.xml* file sample directs JBoss to obtain a username and password by using the HTTP BASIC pop-up. The sample shown from the *jboss-web.xml* file directs JBoss to authenticate that username and password by using the login module configured for the security domain, TEST\_SECURITY\_DOMAIN. You can edit the */opt/UMC/jboss/server/default/conf/login-config.xml* file to change the login module for a particular security domain.

If no login module is defined for TEST\_SECURITY\_DOMAIN, then the “other” security domain is used by default, as shown in this sample from the *login-config.xml* file:

```
<!--
  The default login configuration used by any security domain that
  does not have a application-policy entry with a matching name.
-->
<application-policy name = "other">
  <authentication>
    <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule" flag
    = "required" />
  </authentication>
</application-policy>
```

The *org.jboss.security.auth.spi.UsersRolesLoginModule* login module authenticates usernames and passwords against the *server/default/conf/users.properties* file. The authenticated username must be a member of the role specified in the *web.xml* file. In our example earlier, this is TEST\_ROLE\_NAME.

To provide access to the Web application to user “anonymous” with password “secret” with the *jboss-web.xml* and *web.xml* files shown above, the login module requires the following information:

- From *server/default/conf/users.properties*:

```
anonymous=secret
```

- From *server/default/conf/roles.properties*:

```
anonymous=TEST_ROLE_NAME
```

The following Web applications do not have the *jboss-web.xml* file; you must add the file to provide J2EE-style access control:

- *./licenseServer/adminui/WEB-INF*
- *./prepaid/accountAdmin/WEB-INF*
- *./wkf/webapps/workflow/WEB-INF*

### Default Security Roles for Access Control

Some Web applications have default security roles for access control. For example, *NIC\_Admin* is the default security role for *NIC Web Admin*, and *POM\_Admin* is the default for *Policy Web Admin*. Security roles require configuration for JBoss. For example, to use these roles when JBoss is your Web application server you must add particular information to the *roles.properties* and the *users.properties* files located in the `<jboss-install-dir>/jboss/server/default/conf` directory.

In the *roles.properties* file, you associate users for the roles:

```
nicAdmin=NIC_Admin
pomAdmin=POM_Admin
```

In the *users.properties* file, you associate a password with those users:

```
nicAdmin=<password>
pomAdmin=<password>
```

Suppose you are using the Dynamic Service Activator application. The default security role for Dynamic Service Activator is *DSAAuthorizedClient*. In the *roles.properties* file, you associate users with the role:

```
Elena=DSAAuthorizedClient
Sarah=DSAAuthorizedClient
anonymous=DSAAuthorizedClient
```

In the *users.properties* file, you associate a password with those users:

```
Elena=<password>
Sarah=<password>
anonymous=<password>
```

Other Web application servers may have their own configuration requirements for default security roles. See the documentation for the server that you have deployed for more information.