

Starting and Stopping OC4J

 *OC4J User's Guide*, "Starting and Stopping OC4J," on page 2-6 and "Options for the OC4J Server JAR," on page A-15

Starting OC4J

\$J2EE_HOME is set to \$ORACLE_HOME/j2ee/home.

Execute the following command:

```
java -jar $j2EE_HOME/oc4j.jar <options>
```


Options for this command are not necessary to start OC4J. Use options if you want to exercise more control in starting OC4J.

Stopping OC4J

Execute the following command:

```
java -jar $j2EE_HOME/admin.jar
  ormi://localhost/ <admin>
  <admin-password> -shutdown
```

Deploying a Web Application

 *OC4J User's Guide*, "Quick Start for JSPs and Servlets," on page 2-5 and "Deploying Applications," on page 2-16

Deploy a Web application in one of the following ways:

⇒ **Place JSP pages anywhere in the `j2ee/home/default-web-app` directory.**

The JSP pages are accessible with URLs of the form:

```
http://<apache_host>:7777/j2ee/
<path-to-JSP>
```


For example, a JSP page in `j2ee/home/default-web-app/examples/Hello.jsp` is accessible as `http://<apache_host>:7777/j2ee/examples/Hello.jsp`.

⇒ **Place servlet classes in the `j2ee/home/default-web-app/WEB-INF/classes` subdirectory—in a directory corresponding to their Java package.**

For example, the servlet class `my.HelloServlet` is located in `j2ee/home/default-web-app/WEB-INF/classes/my/HelloServlet.class`. The servlet is accessible through a servlet mapping in the `web.xml` file or from URLs of the form: `http://<apache_host>:7777/j2ee/servlet/<class-name>`. In this case, it is accessible from the following URL:

```
http://<apache_host>:7777/j2ee/servlet/
my/HelloServlet
```

Publishing a Web Module


 *OC4J User's Guide*, "Binding the Web Application," on page 2-17 and "Options for the OC4J Server JAR," on page A-15

Execute the following command to make your J2EE Web application accessible from the OC4J Web server:

```
java -jar admin.jar ormi://<oc4j_host>:
<oc4j_ormi_port> admin <adminusername>
  admin <adminpassword>
  -bindWebApp <app_deploy_name>
  <web_app_name> <web_site_name>
  <context_root>
```

- `<app_deploy_name>` is the application name.
- `<web_app_name>` is the name of the WAR file contained within the EAR file—without the `.WAR` extension.
- `<web_site_name>` is the name of the `web-site.xml` file that denotes the Web site to which this Web application should be bound.
- `<context_root>` is the root context for the Web module.

Undeploying a Web Application


 *OC4J User's Guide*, "Undeploying Web Applications," on page 2-18

Execute the following command:

```
java -jar admin.jar ormi://<oc4j_host>:
<oc4j_ormi_port> admin <adminpassword>
  -undeploy <applicationName> -keepFiles
```

- `<applicationName>` is the name of the Web application that is being undeployed from OC4J.
- `-keepFiles` is the optional switch that prevents application files from being removed.

Setting Up an Emulated Data Source

 *OC4J User's Guide*, "Definition of Data Sources," on page 4-18

A *data source* is a Java object that implements the `javax.sql.DataSource` interface. A data source object is a factory for JDBC connections.

An *emulated data source* (the pre-installed default) wraps around an Oracle data source. An emulated data source is used primarily by applications that access a single database. A *non-emulated data source* is a pure Oracle data source used by applications.


⇒ **Declare an emulated data source in the `data-sources.xml` file as follows, replacing `<oc4j_host>`, `<TTC port>`, and `<DB ID>` with the correct values:**

```
<data-source
  class =
    "com.evermind.sql.DriverManagerDataSource"
  name = "jdbc/DMDSName"
```

```
location = "jdbc/DMDSLocation"
xa-location = "jdbc/DMXADS"
ejb-location = "jdbc/emulatedDS"
username = "scott"
password = "tiger"
url =
  "jdbc:oracle:thin:@<oc4j_host>:
  <TTC port>:<DB ID>"
connection-driver =
  "oracle.jdbc.driver.OracleDriver"
/>
```

- **class attribute:** Any of the Oracle or OC4J data sources.
- **name attribute:** The name of the data source.
- **location attribute:** This must be a unique location, but Oracle recommends that you do not use this location for looking up a data source.
- **xa-location attribute:** This must be a unique location, but Oracle recommends that you do not use this location for looking up a data source.
- **ejb-location attribute:** Indicates the JNDI name used to access a data source for EJBs, JSPs, and servlets. This must be a unique location. Oracle recommends that you use this location for looking up a data source.
- **username and password attributes:** Use when connecting to the database if you want to avoid hardcoding the information in the application code. This is optional.
- **url attribute:** The JDBC connection string for the database.
- **connection-driver attribute:** Any class that implements `java.sql.Driver`.

Using Security

 *OC4J User's Guide*, Chapter 8, "Security"

OC4J security includes authorization, authentication, and confidentiality. The first two of these features are specified in a user repository while confidentiality is handled by the Oracle HTTP Server.

By default, OC4J uses the `XMLUserManager` class and its user repository, `principals.xml`. Because this simple, file-based user repository is not secure, you can specify a user manager with a more secure user repository, such as the `JAZNUserManager` class. The primary purpose of this class is to leverage the JAAS provider as the security infrastructure for OC4J.

Setting Up Authorization Using principals.xml

1. Specify users and groups, as the following XML shows:

```
<principals>
  <groups>
    <group name="allusers">
      <description>Group for all normal
        users</description>
      <permission name="rmi:login" />
      <permission name=
        "com.evermind.server.rmi.RMIPermission"
        />
    </group>
    ...other groups...
  </groups>
  <users>
    <user username="guest"
      password="welcome">
      <description>Guest user</description>
      <group-membership group="allusers" />
    </user>
  </users>
</principals>
```

2. Specify logical roles in a J2EE application.

a. Specify the logical roles that your application uses in the XML deployment descriptors.

Depending on the application type, update one of the following with the logical roles:

- web.xml for a WAR file
- ejb-jar.xml for an EJB JAR file
- application.xml for an EAR file

In each of these deployment descriptors, an XML element known as <security-role> defines the role.

b. In the ejb-jar.xml file, define the bean and method that the role can access, as the following XML illustrates:

```
<method-permission>
  <description>VISITOR role needed for
    CustomerBean methods</description>
  <role-name>VISITOR</role-name>
  <method>
    <ejb-name>customerbean</ejb-name>
    <method-name>*</method-name>
  </method>
</method-permission>
```

c. In the web.xml file, specify the security constraints for a servlet.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>SalesInfo
    </web-resource-name>
    <url-pattern>/salesinfo/*
    </url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-methods>
  <auth-constraint>
    <role-name>VISITOR</role-name>
```

```
</auth-constraint>
</web-resource-collection>
</security-constraint>
```

3. Map logical roles defined in the application deployment descriptors to actual users and groups defined in the principals.xml file.

Specify this mapping in the container-specific deployment descriptor (orion-web.xml, orion-application.xml) with a <security-role-mapping> element, as in the following XML:

```
<security-role-mapping name="VISITOR">
  <group name="allusers" />
</security-role-mapping>
```

The XML maps the logical role VISITOR to the allusers group in the orion-ejb-jar.xml file.

Authentication for HTTP Clients

Most clients to your application are Web browsers, which access the container through the Oracle HTTP Server and mod_ossso. OC4J requests the client to authenticate itself when accessing protected URLs.

Authentication for EJB Clients

When you access EJBs in OC4J, you must pass valid credentials to this server.

- Standalone clients define their credentials in the jndi.properties file deployed with the EAR file. In this case, indicate the username (principal) and password (credentials) to employ when looking up remote EJBs in the jndi.properties file.
- Servlets or JavaBeans running within the container pass their credentials within the InitialContext, which is created to look up the remote EJBs.

