Using this Quick Reference Card

- This quick reference card briefly describes essential procedures for using OC4J. For more detailed descriptions, see the *Oracle9iAS Containers for J2EE Getting Started* HTML file.
- Some sections in this quick reference card supersede the corresponding sections in the *Oracle9iAS Containers for J2EE Getting Started* HTML file.

Basic Installation (Stand Alone)

📖 *Oracle9iAS Containers for J2EE Getting Started* HTML, "Basic Installation"

Once you install Oracle9*i* Application Server, you can find OC4J in the `$ORACLE_HOME/J2EE_containers/oc4j.zip` file.

1. **Make sure a Java2 executable is in the `$PATH`, preferably version 1.3.x.**
2. **Unzip the `oc4j.zip` file in the `$ORACLE_HOME` directory and follow the instructions in the `README.TXT` file.**
3. **Execute the following:**

```
% cd $ORACLE_HOME
% unzip oc4j.zip
% cd $ORACLE_HOME/j2ee/home
% java -jar orion.jar -install
```

   After the install is complete, the `$ORACLE_HOME/j2ee/home` directory contains all the files necessary for running OC4J with a default configuration.

4. **Enter an administration password at the installation prompt.**

   This is used for the administration console command.

Starting and Stopping OC4J

📖 *Oracle9iAS Containers for J2EE Getting Started* HTML, "Starting and Stopping OC4J"

Starting OC4J

Issue the following command:

```
% java -jar $j2ee_home/orion.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

The following command provides a graceful shut down of OC4J:

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

Deploying a Web Application

📖 *Oracle9iAS Containers for J2EE Getting Started* HTML, "Quick Start for JSPs and Servlets" and "Deploying Java Applications"

Deploy a Web application in one of the following ways:

- ➪ **Place JSPs anywhere in the `j2ee/home/default-web-app` directory.** They are accessible with URLs of the form: http://localhost:8888/<path-to-JSP>.

  For example, a JSP in `j2ee/home/default-web-app/examples/Hello.jsp` is accessible as `http://localhost:8888/examples/Hello.jsp`.

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

  For example, the servlet class `my.HelloServlet` is placed in `j2ee/home/default-web-app/my/HelloServlet.class`. The servlet is accessible from URLs of the form: `http://localhost:8888/servlet/<class-name>`. In this case, it would be accessible from the following URL: `http://localhost:8888/servlet/my.HelloServlet`.

If you configured Apache to proxy requests to OC4J, you must add `ProxyPass` commands in the Apache `httpd.conf` file for the URLs giving access to JSPs. The following is a typical example:

```
ProxyPass /examples/ http://<oc4j-host>:8888/
    examples/
ProxyPassReverse /examples/ http://
    <oc4j-host>:8888/examples/
```

Undeploying a Web Application

Issue one of the following commands:

```
java -jar admin.jar ormi://host:rmiport admin
adminpassword -undeploy applicationName
```

-or-

```
java -jar admin.jar ormi://host:rmiport admin
adminpassword -undeploy applicationName
-keepFiles
```

`applicationName` is the name of the Web application being updeployed from OC4J.

`-keepFiles` prevents application files from being removed.

Typical DataSource Setup (Thin Driver)

📖 *Oracle9iAS Containers for J2EE Getting Started* HTML, "Typical Data Source Setup"

J2EE Web applications retrieve connections to the database through `java.sql.DataSource` objects.

1. **Specify the data source in `data-sources.xml`.**
2. **Declare the data source as follows, replacing `<host>`, `<port>`, and `<sid>` with the correct values:**

```
<data-source
class="com.evermind.sql.DriverManagerData-
Source"
connection-driver="oracle.jdbc.driver.Ora-
cleDriver"
ejb-location="jdbc/myPooledDataSource"
location="jdbc/myDataSource"
url="jdbc:oracle:thin:@<host>:<port>:<sid>"
username="scott"
password="tiger"
/>
```

   `class` entry: Always use the "`com.evermind.sql.DriverManagerDataSource`" class.

   `connection-driver` entry: Always use the "`oracle.jdbc.driver.OracleDriver`" class.

   `ejb-location` entry: Indicates the JNDI name used to access a data source, which in this example, uses a pool of JDBC connections. Retrieve a pooled data source from servlets, JSP Pages, Java Beans, or EJBs.

   `location` entry: Indicates the JNDI name to use when you do not want to use a pool of connections.

   `url` entry: The JDBC connection string for the database.

   `username` and `password` entries: Use when connecting to the database if you want to avoid hardcoding them in the application code. This is optional.

Using Security

📖 *Oracle9iAS Containers for J2EE Getting Started* HTML, "Security in OC4J"

OC4J security involves authorization, authentication, and confidentiality.

Setting up authorization

1. **Specify users and groups as described in the following code:**

```
 <principals>
  <groups>
   <group name="allusers">
    <description>Group for all
     normal
    users</description>
    <permission name="rmi:login"
     />
    <permission name="com.eve
     mind.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 type of the application, 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, the roles are defined by an XML element named `<security-role>`.

   b. **Define the bean and method that this role can access as described in the following code:**

```
<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>
```

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 with a `<security-role-mapping>` element as described in the following code:

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

   Here, the logical role `VISITOR` is mapped 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 Apache `mod_sso`. OC4J requests the client to authenticate itself when accessing protected URLs.

Authentication for EJB Clients

When you access EJBs in a *remote* container, you must pass valid credentials to this container.

- Stand-alone 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 use 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.

Setting up Clustering and Load Balancing

📖 *Oracle9iAS Containers for J2EE Getting Started* HTML, "Increasing Performance With Load Balancing and Clustering"

Clustering and load balancing increases performance by redirecting client requests to available OC4J servers.

1. **Install the Web application on all the nodes in your cluster.**
2. **Set up your Web application to replicate its state.**

   On all nodes, edit `orion-web.xml` or `global-web-application.xml <cluster-config/>`
3. **Configure your cluster islands.**

   Edit the `*web-site.xml` file where your Web application is deployed. Modify the `<web-site>` element and add a numerical identifier for the cluster island.
4. **Tell the back-ends about the load balancer for proper redirection.**

   On all nodes, edit `web-site.xml <frontend host="<balancer-host>" port="balancer-port>"`
5. **Distribute your application in the `<app>/WEB-INF/web.xml<distributable/>`**
6. **Start the load balancer with options using `java -jar loadbalancer.jar <options>`**

   Or you can use `load-balancer.xml` if you wish not to use options.

Note: To learn more about servlets, JSPs, and EJBs, go to `http://technet.oracle.com/docs/tech/java/oc4j/content.html` and access the appropriate primer.

📖 Denotes *Oracle9iAS Containers for J2EE Getting Started* HTML