APPENDIX

This appendix lists various scenarios which illustrate the security concepts involved in the Enterprise JavaBeans application. The screen shots contain the output shown in JDeveloper.

I. Successful EJB deployment and initialization of the J2EE server

II. User not defined in target application environment (Application Server)

The following is not defined in the XML-Based provider type jazn-data.xml. Thus the user ‘dan’ will not be authenticated by the server and will be denied access to the EJB application.

```xml
<user>
    <name>dan</name>
    <description>faculty</description>
    <credentials>6NVtuNPSFwkJY7YI/CxzmTcq5q6kHrV</credentials>
</user>
```

1. The EJB client user ‘dan’ attempts to access the bean Student. (*StudentClient: line 68, 69*).
2. The server throws an AuthenticationException saying that the username / password is invalid.
For the user ‘dan’ to be authenticated by the server, the above mentioned XML code has to be defined in the repository *jazn-data.xml*. (The `<credentials>` tag defines the password for user ‘dan’. The password shown above in the XML file has been obfuscated by OC4J.) The actual password for the user ‘dan’ is ‘welcome’. The EJB application has to be redeployed to affect the changes.
III. Declarative Security

The following XML code in `ejb-jar.xml` defines two logical security roles ‘FACULTY’ and ‘STUDENT’. It also defines the method permissions for different methods of the Student bean. Only the role ‘FACULTY’ is allowed to create and remove an instance of the student bean. Both the roles ‘FACULTY’ and ‘STUDENT’ are allowed to access the findByPrimaryKey and findByCourse methods.

```xml
<assembly-descriptor>
  <security-role>
    <role-name>FACULTY</role-name>
  </security-role>
  <security-role>
    <role-name>STUDENT</role-name>
  </security-role>
  <method-permission>
    <description>Only FACULTY is allowed to create and remove a student bean instance</description>
    <role-name>FACULTY</role-name>
    <method>
      <ejb-name>Student</ejb-name>
      <method-name>create</method-name>
    </method>
    <method>
      <ejb-name>Student</ejb-name>
      <method-name>remove</method-name>
    </method>
  </method-permission>
  <method-permission>
    <description>FACULTY and STUDENT roles are allowed to find Student Bean object(s) by using findByPrimaryKey and findByCourse methods</description>
    <role-name>FACULTY</role-name>
    <role-name>STUDENT</role-name>
    <method>
      <ejb-name>Student</ejb-name>
      <method-name>findByPrimaryKey</method-name>
    </method>
    <method>
      <ejb-name>Student</ejb-name>
      <method-name>findByCourse</method-name>
    </method>
  </method-permission>
</assembly-descriptor>
```

The following XML code in `orion-ejb-jar.xml` maps the logical security roles defined in `ejb-jar.xml` to the user/roles of the operational environment (defined in `jazn-data.xml`).

```xml
<assembly-descriptor>
  <security-role-mapping name="STUDENT">
    <user name="joe"/>
  </security-role-mapping>
  <security-role-mapping name="FACULTY">
    <user name="dan"/>
  </security-role-mapping>
</assembly-descriptor>
```

1. The user ‘dan’ (associated with the security role ‘FACULTY’) attempts to create a student bean instance and the instance will persist in the student table in the database. The snapshot below shows the student table to contain only one record before the bean client ‘dan’ attempts to create the bean instance.
J2EE Security and Enterprise JavaBeans

2. User client ‘dan’ is authenticated by the server and has permission to access the EJB methods create (StudentClient: line 28) and findByPrimaryKey (StudentClient: line 29). The snapshot below shows the successful execution of these methods.

3. The snapshot below presents runtime output which shows that the EJB class identifies the caller to be in the ‘FACULTY’ role.

4. The bean instance is created and persistence occurs in the student table in the database.
5. The user ‘joe’ (associated with the security role ‘STUDENT’) can access the EJB method findByPrimaryKey but is not allowed to access the EJB method remove. The snapshot below shows that the user ‘joe’ successfully associates a bean object to the current bean instance by accessing the findByPrimaryKey method (StudentClient: line 29) and loading the bean object with primary key ‘63’ from the Student table in the database. But when the bean client attempts to access the EJB method remove (StudentClient: line 33), the server throws an exception saying that the user ‘joe’ is not allowed to call the EJB method remove.
III. Programmatic Security

1. The snapshot below shows the isCallerInRole(String roleName) method used to check the security role of the caller. *(StudentBean: line 81)*. If the caller is not in the ‘FACULTY’ role the jdbc connection set up in the setEntityContext method of the bean’s lifecycle is closed and the bean life cycle is terminated. Thus any caller who is not in the required security role cannot perform any data access calls with the persistence storage.

2. The snapshot below shows the caller is in the ‘STUDENT’ security role.
3. The caller is thus denied accesses to the EJB create method and the jdbc connection is closed.