2

Installing and Configuring the Runtime Processes

The first step in deploying a J2EE application is setting up the production environment on the appropriate hosts. This involves installing all necessary PowerTier runtime processes and tools, in preparation for deploying your components. If you already have a development installation of PowerTier where you will deploy your application components, you can skip to "Configuring PowerTier Servers" on page 11.

To set up a production system, you must install the required runtime processes on the proper host machines, including any databases and client programs your application will use. To guide you through this process, this chapter contains the following sections:

- Installing the PowerTier Tools
- Configuring PowerTier Servers
- Defining Clusters of Servers
- Configuring the Web Container
- Configuring the Client Container
- Configuring Containers to Use J2EE Services
- Deploying Security Features

Installing the PowerTier Tools

To configure and deploy PowerTier EJB, Web, and client containers, you must install a production version of PowerTier on each node. A production system includes the PowerTier runtime processes listed in Table 1 on page 3 and the following management and configuration tools:

- For managing clusters of PowerTier servers: the Command Center and **ps-agent** (the bootstrap command that handles communication between the Command Center and PowerTier servers).
- For managing and configuring collocated and standalone servlet containers: the administration tools **ps-webadm** and **ps-webgui**.
- For deploying J2EE components: **ps-deploy**, **ps-makeejb**, **ps-makeweb**, and an XML editor, such as XML Spy, for customizing configuration files and deployment descriptors.

To install a production version of PowerTier on each host machine:

1. Start the PowerTier installation program.
2. Select:
   - The PowerTier J2EE application server, which hosts the EJB container, and collocated JSP and servlet engines.

     The installation process sets up a default pantry (the directory that will hold deployed JAR files), and places the location of the pantry in your **CLASSPATH**.
   - The Command Center management agent (**ps-agent**).

     On Windows NT systems, the installation process sets **ps-agent** to run as an NT service. You can specify whether you want this service to run automatically, or whether you want to start it yourself.
   - The PowerTier deployment tools.
   - If your application includes Web components, such as JSPs or servlets, install the Servlet Engine Plug-in for your Web server.

   The installation also includes a Java compiler and a JRE (Java runtime environment).
3. After you install PowerTier, follow the instructions in the *Installation and Configuration Guide* to ensure that the PowerTier J2EE server has been properly configured to work with your databases and database client-side libraries.

The installation process sets up the required directory structure, and places libraries and configuration files in the proper locations. For further instructions, information about licenses, and a list of the directories you will find in your installation, see the *Installation and Configuration Guide* and the *Web Application Development Guide*.

Configuring PowerTier Servers

You use the Command Center's Web interface to configure most aspects of PowerTier J2EE servers and clusters of servers. To configure J2EE servers, you need the following information from your production environment:

- Resource locations (naming service, URLs, etc.)
- EJB references
- Environment entries
- Security information, as listed in "Deploying Security Features" on page 22

This section describes basic server configuration in the following topics:

- Server Configuration Files
- Using the Command Center to Configure Servers
- Defining Servers
- Defining Clusters of Servers

Server Configuration Files

PowerTier servers use the following configuration files:

- the server configuration (**.ptc**) file

  The PowerTier tools **ps-gen** and **ps-makeejb** generate initial server configuration files, using the name of the project: ***projectName*.ptc**.
- the container configuration file (**pt-config.xml**)

  If your application uses J2EE services, you need both a **.ptc** file and **pt-config.xml**. For more information, see "Configuring Containers to Use J2EE Services" on page 19.
- the Web server plug-in configuration file (**ps-pi.conf**)
- the servlet container properties file (**se.properties**)

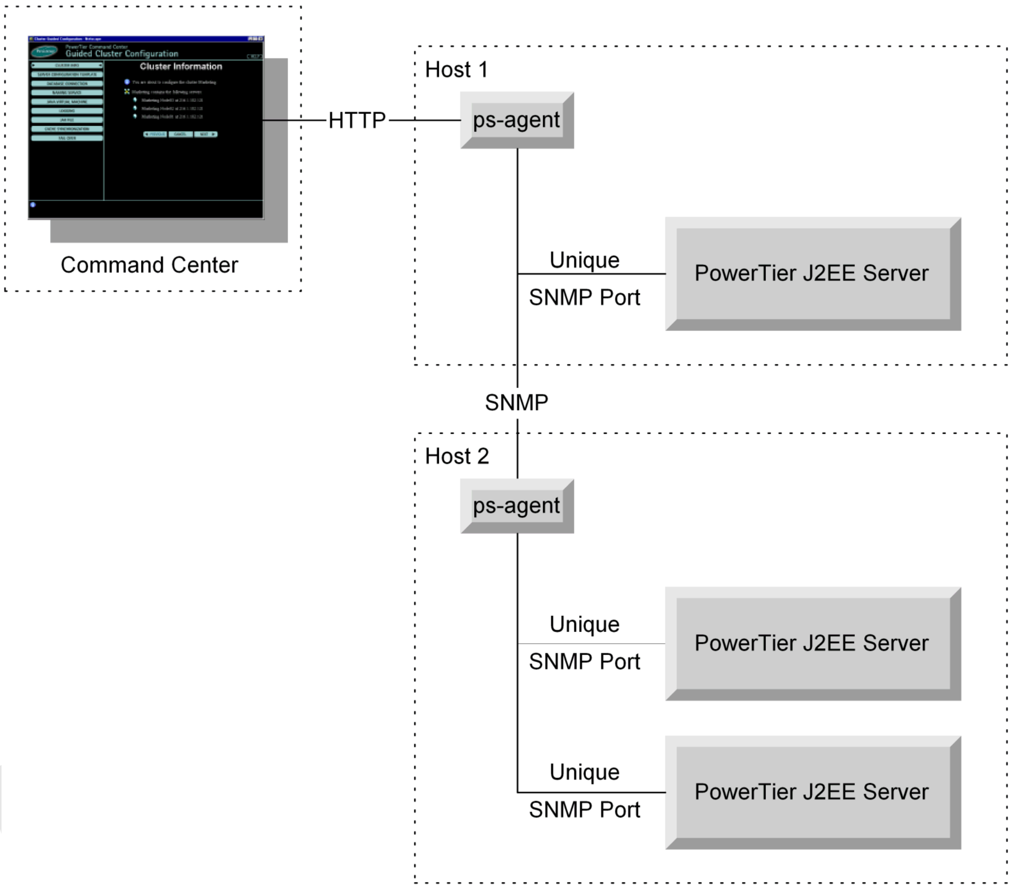
When you configure PowerTier J2EE servers, the Command Center saves the configuration information in a separate **.ptc** file for each server (***serverName*.ptc**), in the **config** directory of the PowerTier installation on that server's host. If necessary, during development you can edit the **.ptc** file yourself to make minor changes, but Persistence recommends that you use the Command Center whenever possible. For more information about the format of the **.ptc** file, see the *Reference Guide*.

You can run the Command Center on one host machine to configure local or remote servers in your network. If your developers have provided an initial **.ptc** file, you must place this file in the **config** directory of the PowerTier installation on the machine where you will run the Command Center.

Using the Command Center to Configure Servers

The Command Center uses SNMP (simple network management protocol) to communicate with PowerTier servers. Each individual server must have a unique SNMP port number. When you group PowerTier servers in a cluster, each member server of that cluster uses the same SNMP port. When you configure PowerTier servers, you specify the SNMP port. Figure 5 shows how the Command Center communicates with PowerTier servers.

Figure 5. Command Center Connections to PowerTier Servers



You must start **ps-agent** on each host computer that contains one or more PowerTier J2EE servers. Each **ps-agent** communicates with the others, allowing you to use a single Command Center to configure and control all of the PowerTier J2EE servers.

To start the Command Center:

1. If you did not set up **ps-agent** to run automatically (during the installation process), you must start it before running the Command Center. To start **ps-agent**, do one of the following:
   - On Windows NT, open the **Control Panel** and then open the **Services** dialog. Select **PowerTier Agent** and then click **Start**.

- On either UNIX or Windows, enter the following command:

  ```
  ps-agent -f configFile -p port
  ```

  The default configuration file is **ps-agent.ini**. The default port is **9080**.

When **ps-agent** starts, it creates a list of the **.ptc** files in the **config** directory. As **ps-agent** runs, it uses this list to determine which servers the Command Center will manage. Do not change or delete any of these files while either **ps-agent** or the Command Center is running.

2. Open a browser window.
3. Enter the URL `http://hostName:portName` to start the Command Center.

   Use the same ***portName*** you used when you started **ps-agent**.

Defining Servers

For eachPowerTier server in your system, you must provide the following information:

- The server's name, the host name, and a description to identify that server.
- Database connection pools, including the name, database type, and login information.
- Naming service information, including the naming service your applications uses and the location of the ORB. By default, PowerTier uses COSNaming, but you can also use a JNDI-compliant implementation of LDAP. For examples of using LDAP, see the *PowerTier Server and EJB Development Guide* and the *Client Development Guide*.

Note: If you use COSNaming, you can use **ps-tnameserv** during unit testing, but for production, Persistence recommends that you use **jorbd** or LDAP. The **jorbd** naming service is persistent, in contrast to **ps-tnameserv**, which is a transient naming service. If the **jorbd** naming service fails, and you have configured it to do so, another instance automatically starts and continues serving client requests.

- JAR files for this server to load, including the name of each JAR file, a description to identify that file, and the connection pool to use for the components in that JAR.

You can also use the Command Center to specify additional information about your PowerTier servers, including:

- Java Virtual Machine information, including the version and the path (if you use a JVM other than the default one supplied with PowerTier).

- Cache settings, including the overall cache-clearing policy, the number of hash buckets in the shared cache, and the number of hash buckets in the transactional cache. For recommendations, see "Configuring the Cache" on page 46.
- Whether you want to enable logging of server messages. For more information, see "Monitoring Cache Usage" on page 48.
- EJB components in your application, including:
  - the names of your entity beans
  - the enterprise beans in each JAR file
  - cache-clearing policies and database connections pools for each bean

After you configure your servers, you will want to test your configurations. For instructions, see "Starting PowerTier Servers and Clusters of Servers" on page 38.

Defining Clusters of Servers

You can use the Command Center to combine PowerTier servers into clusters for load balancing, cache synchronization, failover, and failback. When you configure a cluster, you configure all of its member servers at the same time. To perform this configuration, you need to know how your application implements these features. For detailed information about load balancing, cache synchronization, failover, and failback, see the *PowerTier Server and EJB Development Guide*.

When you assign a server to a cluster, the Command Center prefixes the name of the cluster to the **ServerName** element in the member server's **.ptc** file – for example, **Bank.HomeOffice** indicates the **HomeOffice** server in the **Bank** cluster.

To define a cluster of servers, you provide the following information:

- The name of the cluster, and the member servers in the cluster.
- For each member server: the name of the server, its host name or IP address, and an optional description of that server.
- Cache settings, database connection pools, naming service, and JVM information for the cluster as a whole, and optionally for individual servers.

  If your application requires cache-clearing access to the naming service, you might need a more robust naming service than **jorbd**, such as LDAP. The Netscape Directory Server provides LDAP load balancing and failover across machines.
- Cache synchronization information, including which messaging system your application uses and what classes your application uses to implement cache synchronization.
- Whether to enable transparent failover and failback for this cluster and what classes your application uses to implement failover.

For full instructions and explanations of the cluster configuration values you must specify, see the *Command Center Guide*.

Configuring the Web Container

If your application uses the PowerTier Servlet Engine, make sure you install the Web server plug-in for your Web server. The combination of the PowerTier Servlet Engine, the Web server plug-in, and a Web server is known as the *Web container*.

To configure the Web container, you use the **ps-webadm** command (or, if you prefer a graphical interface, the **ps-webgui** command). If you run your servlet containers collocated with a PowerTier J2EE server, you can use **ps-webadm** to install or remove a Web application to or from a servlet container, and to stop a running servlet container or delete a servlet container from your system.

You can deploy Web components, such as JSPs and Java servlets, as bundled Web applications (packaged in WAR files), or in an open directory structure, as shown in Example 3 on page 32. Deploying Web applications to multiple standalone servlet containers automatically configures the Web server plug-in to support load balancing and failover across those servlet containers.

If you use the servlet container in-process with the PowerTier J2EE server, you must provide the PowerTier server with information about the collocated servlet containers. To do this, you must add a **JSPServletEngine** element for each servlet container to the **.ptc** file.

For example:

```
<JSPServletEngine
    EngineType="ServletMill"
    EngineID="D:\PowerTierHome\web\se\se"
    StopTimeout="60"
</JSPServletEngine>
```

Installing Web Components to Servlet Containers

To install a Web application into the PowerTier Web container, you use either **ps-webadm** or **ps-webgui**. Both administration tools do the same things; you can choose which interface you prefer:

- The **ps-webadm** command provides an ASCII-based interface for configuring and running Web applications. You can also bypass the ASCII interface and specify some options directly on the command line.

- The **ps-webgui** command provides a graphical user interface for configuring and running Web applications.

For information about using these tools, see the *Web Application Development Guide*.

To use Web components, you must install them to both a servlet container and the Web server plug-in, as follows:

- To install a Web application in a servlet container, use the **ps-webadm** command, as follows:

  ```
  ps-webadm -install ptWarFile [-contextroot contextRoot]
      -se [name | location]
  ```

  When you use this command, the administration tool does the following:

  a. Copies the PowerTier WAR file to the Web pantry (if it is not already there) and expands it (if it is not already expanded).

  b. Modifies the servlet container's properties file (**se.properties**) to support this Web application.

  c. Modifies the **MILLSE** directive in the Web server plug-in's configuration file (**ps-pi.conf**) to inform the plug-in about the installed application.

- To install a Web application in the Web server plug-in, use the following command:

  ```
  ps-webadm -install ptWarFile [-contextroot contextRoot] -pi [docroot]
  ```

  When you use this command, the administration tool does the following:

  a. Copies the PowerTier WAR file to the Web pantry (if it is not already there) and expands it (if it is not already expanded).

  b. Copies all Web components that are not in the **WEB-INF** or **META-INF** directory to the Web server's document root directory.

  c. Adds **MillMount** directives to the plug-in's configuration file (**ps-pi.conf**) for the specified Web application.

Configuring the Client Container

The client container uses the PowerTier configuration file (**pt-config.xml**). This file contains sections for each type of container: EJB, client, and Web. In addition, it contains sections where you configure J2EE services, such as JDBC, JMS, and JavaMail. For more information, see "Configuring Containers to Use J2EE Services" on page 19.

In the client portion (**pt-client**) of the **pt-config.xml** file, you specify the JAR files that contain your Java client programs.

When the **ps-deploy** command extracts application components from EAR and JAR files, it creates a default **pt-config.xml** file with commented-out entries. You can modify this file to provide information about your application's environment. For a complete description of the **pt-config.xml** file, see the *Reference Guide*.

If you use COSNaming, you must install the ORB on each client machine. To install the ORB:

1. Copy the file **ps-ejb-client.jar** file from the **component\client\lib** directory of your PowerTier installation to any directory on your client program's host computer.
2. Add the location of the **ps-ejb-client.jar** file to the **CLASSPATH** for the client program.

To run your client program using the PowerTier client container, you must tell the client container where to find your client JAR files.

To specify client JAR files:

1. Open the **pt-config.xml** file in an XML editor.
2. In the **pt-client** section, add **jar-file** elements with the names of your files. For example:

```
<pt-client>
    <jar-files>
        <jar-file> BankClient.jar </jar-file>
        <jar-file> bank-client.jar </jar-file>
            ...
    </jar-files>
</pt-client>
```

Configuring Containers to Use J2EE Services

PowerTier supports the following J2EE services for EJB, Web, and client containers:

- JDBC – provides access to relational databases.
- JMS – provides a reliable message-passing service.
- JavaMail – allows J2EE applications to send e-mail messages.
- URL resource references – allow an application to use a hard-coded alias for a URL that is mapped to an actual URL at runtime.

You specify information about the J2EE services your application uses in the service-specific sections of the **pt-config.xml** file. If you use J2EE services, you must add a **config-file** element to the **.ptc** file to specify the location of your **pt-config.xml** file.

Using JDBC

To specify how your application uses JDBC, you must provide the locations of your databases and any necessary login information. You also configure any connection pools you want your application to use.

To use JDBC with a J2EE application, add **datasource-def** entries for each of your databases to the server configuration, in one of two ways:

- Use the Command Center to specify database definitions in the **.ptc** file.
- Edit the **pt-config.xml** file.

  Database definitions in the **pt-config.xml** file take precedence over corresponding entries in the **.ptc** file.

The following example can appear in either the **.ptc** file or the **pt-config.xml** file:

```
<datasource-defs>
    <datasource-def>
        <res-ref-id> ConfigureTable_DataSource_Id </res-ref-id>
        <driver> oracle.jdbc.driver.OracleDriver </driver>
        <url> jdbc:oracle:thin:@charger:1521:pstest </url>
        <user-name> joe </user-name>
        <password> mypassword </password>
        <pooling-enabled> True </pooling-enabled>
        <min-conn> 1 </min-conn>
        <max-conn> 5 </max-conn>
        <login-timeout> 10 </login-timeout>
        <idle-timeout> 30 </idle-timeout>
        <checkout-timeout> 120 </checkout-timeout>
    </datasource-def>
</datasource-defs>
```

Using JMS

To allow your application to use JMS, you provide information about the names and locations of JMS message queues, the names of your Resource Manager Connection Factory objects, login information to obtain connections, and any provider-specific properties you need to set.

To use JMS with a J2EE application, add **jms-connection-factory-def** entries to the **pt-config.xml** file. For example:

```
<jms-connection-factory-defs>
    <jms-connection-factory-def>
        <res-ref-id> MyJMS_JMSQueueConnectionFactory_Id </res-ref-id>
        <provider> SpiritMessenger </provider>
        <user-name> tina </user-name>
        <password> whatever </password>
        <properties>
            <property>
                <name> messageChannel </name>
                <value> stream://padres:16789 </value>
            </property>
            <property>
                <name> providerProperty </name>
                <value> propValue </value>
            </property>
        </properties>
    </jms-connection-factory-def>
</jms-connection-factory-defs>

<jms-destination-defs>
    <jms-destination-def>
        <res-ref-id> DestId1 </res-ref-id>
        <destination-name> TestTopic </destination-name>
    </jms-destination-def>
</jms-destination-defs>
```

Using JavaMail

To allow your application to use JavaMail, you provide information about your Resource Manager Connection Factory objects, login information to obtain mail sessions, and provider-specific properties, such as your mail host and what protocol it uses.

To use JavaMail with a J2EE application, add **mail-session-def** entries to the **pt-config.xml** file. For example:

```
<mail-session-defs>
    <mail-session-def>
        <res-ref-id> MyJavaMail_JavaMailSession_Id </res-ref-id>
        <user-name> tina </user-name>
        <password> whatever </password>
        <properties>
            <property>
                <name> mail.store.protocol </name>
                <value> SMTP </value>
            </property>
            <property>
                <name> mail.transport.protocol </name>
                <value> com.acme.SMTPTRANSPORT </value>
            </property>
            <property>
```

```
                <name> mail.host </name>
                <value> charger </value>
            </property>
            <property>
                <name> mail.user </name>
                <value> tina </value>
            </property>
            <property>
                <name> mail.from </name>
                <value> tina@charger </value>
            </property>
            <property>
                <name> mail.debug </name>
                <value> True </value>
            </property>
        </properties>
    </mail-session-def>
</mail-session-defs>
```

Using URL Resource References

At development time, you may not know the actual URL that represents the location of all of your application resources. You can use URL references to provide an alias that can be mapped to the actual location at deployment time.

To specify how your application uses URL aliases, you provide information about your Resource Manager Connection Factory objects and the URLs you want to map.

To use URL aliases with a J2EE application, add **url-def** entries to the **pt-config.xml** file. For example:

```
<url-defs>
    <url-def>
        <res-ref-id> MyURL_URLSession_Id </res-ref-id>
        <url> http://localhost:8000/index.html </url>
    </url-def>
</url-defs>
```

Deploying Security Features

To deploy a secure application, you must complete the following security tasks:

- Creating and Distributing Certificates

  This includes the following tasks:

  - installing the CA on a secure host computer
  - creating the CA's trusted certificate
  - creating a certificate for each PowerTier server's host computer

- Configuring the Server for Security

  You provide information about your application's security policy definitions (authentication, encryption, and port numbers) and the classes used to implement security functions.You can use the Command Center to do this or you can edit the **.ptc** file directly.

- Protecting Enterprise Beans at Deployment

  This includes the following tasks:

  - Defining security roles: you declare all security roles when deploying your server; you need not compile any security role information into your application.
  - Modifying deployment descriptors to specify security information for JAR and WAR files, individual enterprise beans, and Web application resources.

- Providing Client-Program Security

  This includes the following tasks:

  - Providing the trusted CA certificate file, so your client program can validate the digital signature on the PowerTier server's certificate.
  - For Web clients, using your Web server's SSL (or equivalent features) to protect servlets.

- Identifying Users

  This includes the following tasks:

  - Adding user information to the user-accounts file and the access-control list.
  - Associating security roles with users in the user-role mapping file.

For specific instructions for each of these tasks, consult the *Security Guide* and the *Web Application Development Guide*.

3

Configuring and Packaging Application Components

Once you have installed the necessary runtime processes on your production hosts, you can install and configure the application components. This involves unpackaging EAR, JAR, and WAR files; modifying deployment descriptors; and repackaging your components for deployment.

This section contains the following topics that describe deploying application components:

- J2EE Application Structure and Deployment Tools
- Creating the Application Directory Structure (Unpackaging)
- Deploying EJB Components
- Deploying Web Components
- Deploying Client Programs
- Configuring PowerTier Containers to Use Your Components

J2EE Application Structure and Deployment Tools

Each type of component in a J2EE application is packaged in its own type of archive file. In addition, each type of component requires its own deployment descriptors. Table 2 shows the types of archive files and the associated deployment descriptors.

Table 2. J2EE Component Packaging

Component Type	Archive	Deployment Descriptors
Entire J2EE application	EAR	**application.xml** (J2EE-standard)
EJBs	JAR	**ejb-jar.xml** (J2EE-standard) **pt-jar.xml** (PowerTier-specific)
Web components	WAR	**war.xml** (J2EE-standard) **ptwar.xml** (PowerTier-specific)
Java client programs	JAR	**application-client.xml** (J2EE-standard) **pt-application-client.xml** (PowerTier-specific)

Note: The PowerTier server does not recognize EAR files in this release. Therefore, you must package EJB components and Java client programs into JAR files, and Web components into WAR files to work with PowerTier containers.

PowerTier provides the following deployment tools to work with J2EE application components:

- **ps-deploy** extracts the contents of EAR, JAR, and WAR files and creates or updates deployment descriptors for the extracted components.
- **ps-makeejb** compiles and packages EJB components and Java client code for deployment.
- **ps-makeweb** builds and packages Web components for deployment.

You use **ps-deploy** when you have application components packaged in J2EE archive files. Often these components come from other development groups, and can even come from other vendors. When you receive these files, you need to extract their contents to customize the deployment descriptors and configuration files to reflect your runtime environment.

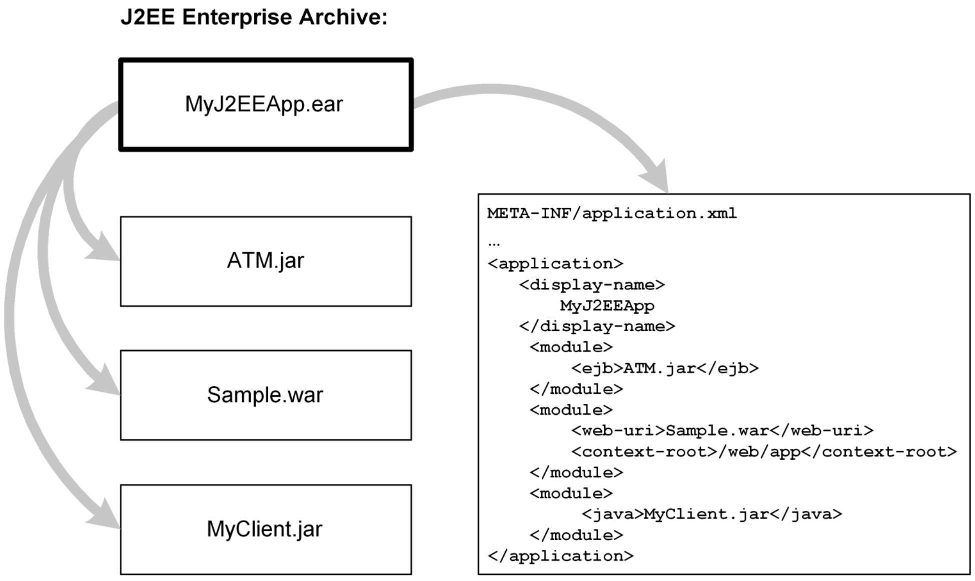
The **ps-deploy** command enables you to place both EJB and Web components in a PowerTier-specific directory structure for deployment. The **ps-deploy** command uses the **ps-makeejb** and **ps-makeweb** commands to create or update entries in deployment descriptors and configuration files based on the elements extracted from JAR or WAR files. You can also use these tools to update and repackage modified application components.

Creating the Application Directory Structure (Unpackaging)

When you receive an EAR file from a bean provider, you must extract the files from the archived modules, modify the deployment descriptors, generate container-adaptor code (if it has not already been generated), and re-package your components for deployment.

Figure 6 shows the internal structure of a sample EAR file.

Figure 6. J2EE Enterprise Application Structure



```
META-INF/application.xml
...
<application>
   <display-name>
       MyJ2EEApp
   </display-name>
    <module>
        <ejb>ATM.jar</ejb>
    </module>
    <module>
        <web-uri>Sample.war</web-uri>
        <context-root>/web/app</context-root>
    </module>
    <module>
         <java>MyClient.jar</java>
    </module>
</application>
```

To extract the components of this EAR file to a standard directory structure:

1. Enter the following command:

   ```
   ps-deploy MyJ2EEApp.ear
   ```

   The **ps-deploy** command creates separate project directories for each module listed in the **application.xml** descriptor file, as shown in Example 1.

2. To specify a root directory other than the current directory for the extracted files, use **ps-deploy** with the **-outputDir** option; for example:

```
ps-deploy MyJ2EEApp.ear -outputDir C:\J2EEApps\MyAppRoot
```

Example 1. Files Extracted from an Enterprise Archive

```
MyJ2EEApp\                      → current directory (application root)
    application.xml             → application DD extracted from EAR file
    ejb\                        → generated repository for EJB modules
        ATM\                    → generated project directory for an EJB module
            ejb-jar.xml         → EJB DD extracted from JAR file
            pt-jar.xml          → PowerTier DD generated by ps-deploy
            ps-makeejb.cfg      → generated EJB project configuration file
            PSInternal\         → generated directory for JAR contents
                …\*.class
                META-INF\
                    ejb-jar.xml

    java\                           → repository for Java modules in EAR file
        MyClient\                   → generated Java module project directory
            application-client.xml→ application client DD extracted from JAR
            pt-application-client.xml→ PT client DD generated by ps-deploy
            PSInternal\             → generated directory for JAR contents
                …\*.class
                META-INF\
                    application-client.xml
                    MANIFEST.MF

    web\                        → generated directory for Java modules
        Sample\                 → project directory generated for a Web module
            *.jsp               → Web application files
            *.html              → static HTML pages
            images\*.gif        → image files
            *.class             → client-side applets, beans, classes
            WEB-INF\
                web.xml             → standard Web deployment descriptor
                ptwar.xml           → PowerTier Web deployment descriptor
                lib\*.jar           → extracted JAR files
                classes\…\*.class → servlets and helper classes

    server\
        pt-config.xml           → generated PowerTier configuration file
        MyJ2EEApp.ptc           → generated PowerTier server configuration file
```

If you want to extract the application components into the standard directory structure without creating or updating deployment descriptors, you can use **ps-deploy** with the **-extractOnly** option. If you do this, you can use the **ps-makeejb** and **ps-makeweb** commands to create deployment descriptors for your EJB, Web, and Java client components, as described in the following sections.

Deploying EJB Components

EJB components require two deployment descriptors: **ejb-jar.xml** and **pt-jar.xml** if you are deploying them in a PowerTier container. When you extract the contents of an EJB JAR file, **ps-deploy** creates default versions of these descriptors, or updates existing ones, based on the components it extracts.

Modifying deployment descriptors can include resolving external dependencies, and replacing the default values in commented-out elements. When you make any changes to deployment descriptors, Persistence recommends that you use the **-validateDD** option of **ps-makeejb** before repackaging your components.

To deploy EJB components:

1. If you do not have deployment descriptors, use the following command to create them:

   ```
   ps-makeejb -createDD
   ```

1. To deploy components extracted from a JAR file, verify the following information in the deployment descriptors, and edit the descriptor files if necessary:

   - Resource locations (naming service, URLS, etc.)

     resource-ref entries in the **ejb-jar.xml** file specify the resource manager connection factory (RMCF) references in your application. Corresponding **resource-ref** entries in the **pt-jar.xml** file include additional PowerTier-specific information about the resources your application uses. For more information, see "Using J2EE Services with EJB Components" on page 31.

   - EJB references

     ejb-ref entries in the **ejb-jar.xml** file specify the names of enterprise beans that your entity and session beans need to reference.

   - Environment entries

     env-entry elements in the **ejb-jar.xml** file specify environment values for your enterprise beans.

- Security

  If you are deploying third-party beans in a secure application, you must include them in **secure-resource-def** entries in the **pt-jar.xml** file.

  For information about modifying specific sections of the deployment descriptors, see the *PowerTier Server and EJB Development Guide*. For details of deployment descriptor syntax, see the *Reference Guide*.

2. To validate your changes, run the following command:

   ```
   ps-makeejb -validateDD
   ```

3. To generate container-adapter code and package your EJB components into a PowerTier-specific JAR, use the following command:

   ```
   ps-makeejb -ptJar MyEjbJar
   ```

   This command does the following things:

   a. Generates the remote and home interface implementations and places them in the proper locations.

   b. Generates the RMI stubs and skeletons.

   c. Creates a new PowerTier JAR file; in this case, ***MyEjbJar*-pt.jar**.

4. Copy the PowerTier-specific JAR file to the Persistence pantry for further testing or deployment.

5. To test your application, start the server, as described in "Starting PowerTier Servers and Clusters of Servers" on page 38.

Example 2 shows the project directory after you run **ps-makeejb -ptJar** for a project called **ATM**.

Example 2. EJB Component Directory After ps-makeejb

```
ejb\                     → generated repository for EJB modules
    ATM\                 → generated project directory for an EJB module
        ejb-jar.xml      → EJB DD extracted by ps-deploy
        pt-jar.xml       → PowerTier DD generated by ps-makeejb -createDD
        ps-makeejb.cfg   → configuration file generated by ps-deploy
        ATM.jar          → generated by ps-makeejb -ejbJar
        ATM-pt.jar       → generated by ps-makeejb -ptJar
        ATM-client.jar   → generated by ps-makeejb -clientJar
        PSInternal\      → generated by ps-deploy, now including PSImpl
            …\*.class    →  classes and RMI stubs and skeletons
            META-INF\
                ejb-jar.xml
```

Using J2EE Services with EJB Components

If your EJB components use J2EE services, such as JDBC or JavaMail, you must specify this in the EJB deployment descriptors, as follows:

1. To use JDBC, add **resource-ref** entries to the **ejb-jar.xml** and **pt-jar.xml** deployment descriptors.
2. To use JMS or URL aliases, add **resource-reference-def** entries to the **pt-jar.xml** deployment descriptor.
3. If your application uses JavaMail, implemented with session beans, add **resource-ref** entries to **session** elements for those session beans in the **ejb-jar.xml** deployment descriptor.

For examples, see the *PowerTier Server and EJB Development Guide*.

Deploying Web Components

A PowerTier Web application has the following structure:

```
webAppDir\
    *.jsp                → Web application files
    *.html               → static HTML pages
    images\*.gif         → image files
    *.class              → client-side applets, beans, classes
    WEB-INF\
        web.xml          → standard Web deployment descriptor
        ptwar.xml        → PowerTier Web deployment descriptor
        lib\*.jar        → libraries in JAR files
        classes\…\*.class → servlets and helper classes
```

When you use the **ps-deploy** command to extract Web components from an EAR or WAR file, it places the components in the proper places in this structure. You can use the **ps-makeweb** command to create this structure from files of your own. For details, see the *Web Application Development Guide*.

Web components require two deployment descriptors: **web.xml** and **ptwar.xml** if you are deploying them in PowerTier servlet and JSP containers. When you extract the contents of a WAR file, **ps-deploy** calls **ps-makeweb** to create default versions of these descriptors, or update existing ones, based on the extracted components. Unless you specify otherwise (using the **-noValidate** option), **ps-makeweb** validates the contents of your deployment descriptors against the corresponding DTDs (document type definitions).

You deploy PowerTier Web components to the Web pantry. Both standalone and collocated servlet containers use this location. The **%PERSISTENCE_WEB_PANTRY%** environment variable specifies the default Web pantry. If this variable is not defined, the Web container uses the **web\apps** subdirectory of your PowerTier installation.

Note: By default, **%PERSISTENCE_WEB_PANTRY%** is not defined. For security reasons, it is important that you do not include the Web pantry in the **CLASSPATH**.

To deploy Web components:

1. If you do not have deployment descriptors, use the following command to create them:
   ```
   ps-makeweb -all
   ```
2. Use the following command to update **servlet** and **servlet-mapping** elements for the precompiled servlets in your Web application directory:
   ```
   ps-makeweb -updateDD
   ```
3. In each project-specific directory, edit the **web.xml** and **ptwar.xml** files as needed.
   For examples of modifying specific sections of the deployment descriptors, see the *Web Application Development Guide*. For details of deployment descriptor syntax, see the *Reference Guide*.
4. To repackage the Web application and copy the files to the Web pantry, use the following command:
   ```
   ps-makeweb -all
   ```
   If you prefer, you can deploy your Web application using an “open” directory structure (as shown in Example 3), rather than repackaging it into a WAR file.
5. To install this Web application into a particular servlet container, run **ps-webadm** or **ps-webgui**. For further instructions, see “Installing Web Components to Servlet Containers” on page 16.

Example 3 shows the project directory after you run **ps-makeweb -all** on a Web application called **Sample**.

Example 3. Web Components Directory After ps-deploy and ps-makeweb

```
web\                     → generated directory by ps-deploy
    Sample\              → project directory generated by ps-deploy
    Sample-pt.war        → WAR file generated by ps-makeweb -all
        *.jsp            → Web application files
        *.html           → static HTML pages
        images\*.gif     → image files
        *.class          → client-side applets, beans, classes
```

```
WEB-INF\
    web.xml           → standard Web deployment descriptor
    ptwar.xml         → PowerTier Web deployment descriptor
    lib\*.jar         → extracted JAR files
    classes\…\*.class → servlets and helper classes
```

Using J2EE Services with Web Components

If your Web components use J2EE services, such as JDBC or JavaMail, you must add **resource-ref** entries for each service to the **web.xml** and **ptwar.xml** deployment descriptors. For examples, see the *Web Application Development Guide*.

Deploying Client Programs

Standalone Java client programs require two deployment descriptors: **application-client.xml** and **pt-application-client.xml** if you are deploying them in a PowerTier container. When you extract Java client components from a JAR file, **ps-deploy** creates default versions of these descriptors, or updates existing ones, based on the components it extracts.

After you modify the generated deployment descriptors, Persistence recommends that you use the **-validateClientDD** option of **ps-makeejb** before repackaging your components.

To deploy Java client programs:

1. If you do not have deployment descriptors, use the following command to create them:

   ```
   ps-makeejb -createClientDD
   ```

   This command creates default deployment descriptors, and a default container configuration file (**pt-config.xml**) if you do not already have one.

2. Using an XML editor, make any necessary changes to the deployment descriptor files. For example, you might need to include entries to specify Application Naming Environment settings.

   For examples of modifying specific sections of the deployment descriptors, see the *Client Development Guide*. For details of deployment descriptor syntax, see the *Reference Guide*.

3. If you modify the deployment descriptors or the container configuration file, use the following command to validate your changes:

   ```
   ps-makeejb -validateClientDD [-ptConfigXml My-pt-config.xml]
   ```

   If you have a **pt-config.xml** file, specify it here to include it in the validation process.

4. If your client program does not include an XML parser, use the following command to serialize the client deployment descriptor files and the container configuration file:

   ```
   ps-makeejb -serializeClientDD
   ```

   This command produces a serialized deployment descriptor that combines both deployment descriptors (**application-client.xml** and **pt-application-client.xml**) and your PowerTier configuration file (**pt-config.xml**) into a single file, with the default name **application-client.ser**. If you serialize your deployment descriptors, you must use the **ps.client.descriptor** property value to specify the location of the **.ser** file when deploying your application client.

5. To package your application client into a PowerTier JAR file, use the following command:

   ```
   ps-makeejb -clientJar
   ```

   This command creates a client JAR file called ***project*-client.jar**, which contains:

   - The class that contains the client's main method.
   - Any additional classes used by the client.
   - Any other resources used by the client (such as images in **gif** or **jpeg** format).
   - The **application-client.xml** deployment descriptor.
   - The JAR file's manifest (**MANIFEST.MF**), which contains an entry that identifies the client's main class.

6. Copy the client JAR file to a location in your **CLASSPATH**, or to the pantry directory specified in your **pt-config.xml** file.

7. To test your application client, start the PowerTier client container, as described in "Starting Java Client Programs" on page 41.

Example 4 shows the client directory for an application called **MyClient** after you run **ps-makeejb** with the options listed.

Example 4. Java Client Directory After ps-makeejb

```
java\                           → repository for Java modules in EAR file
    MyClient\                   → generated by ps-deploy
        application-client.xml  → from ps-makeejb -createClientDD
        pt-application-client.xml→ from ps-makeejb -createClientDD
        pt-config.xml           → from ps-makeejb -createClientDD
        application-client.ser  → from ps-makeejb -serializeClientDD
                                →  (optional)
```

```
PSInternal\                 → generated by ps-deploy
    …\*.class
    META-INF\
        application-client.xml
        MANIFEST.MF
```

Using J2EE Services with Client Programs

If your client program uses J2EE services, such as JDBC or JavaMail, you must specify this in the client deployment descriptors, as follows:

1. To use JDBC, add **resource-ref** entries to the deployment descriptors **application-client.xml** and **pt-application-client.xml**.
2. To use JMS, JavaMail, or URL aliases, Add **resource-ref** entries for each service to the **application-client.xml** descriptor.

For examples, see the *Client Development Guide*.

Configuring PowerTier Containers to Use Your Components

When your application components are ready for deployment, you must tell the PowerTier containers where to find the components. To do this, you modify the configuration files for each container. If the bean provider already specified this information, you might not have to do any further configuration at this point.

Related Information

See the following sources for related information:

Topic	Location
Configuring PowerTier containers	Chapter 2, "Installing and Configuring the Runtime Processes"
Running the PowerTier containers	Chapter 4, "Starting the Runtime Processes"