



WebSphere Application Server V7: Deploying Applications

In Chapter 14, *Packaging Applications for Deployment*, we discuss how to use the Rational® Application Developer Assembly and Deploy Features for WebSphere® 7.0 (RAD-AD) to perform common tasks for packaging an application.

In this chapter, we show you how to deploy the application. We take you through setting up the environment for the application, and then deploying the application itself. Next, we explain how to deploy the client part of the application. The deployment tasks in this chapter can also be automated using command-line tools, as explained in Chapter 8, *Administration with scripting*.

We cover the following topics:

- ▶ Preparing the environment
- ▶ Deploying the application
- ▶ Deploying application clients
- ▶ Updating applications

Note: The application that we prepared is the ITSO Bank application, which was developed by the team who wrote the *Rational Application Developer V7.5 Programming Guide*, SG24-7672, Redbooks® publication. The application can be downloaded from the Additional Materials section in that book, available at:

<http://www.redbooks.ibm.com/redpieces/abstracts/sg247672.html?Open>

The Additional Materials section contains two ZIP files. To prepare for this chapter, download the `7672code.zip` file and unpack it to a directory on your computer. The database directory in this ZIP file contains scripts that we will use to prepare the database for the application.

If you are working on a pre-JEE 5 application or are using EJB 2.1, or earlier, modules, also refer to Chapter 14 in the *WebSphere Application Server v6.1*, SG24-7304, Redbooks publication for specific details on earlier versions, available at:

<http://www.redbooks.ibm.com/abstracts/sg247304.html?Open>

Preparing the environment

In this chapter, we show you how to set up a fairly complete environment for the ITSO Bank application and deploy the EAR file. You will not always need or want to customize the environment as extensively as we do in this chapter. Some steps are optional. If all you want to do is deploy your application quickly, using the WebSphere defaults for directory names, log files, and so forth, skip to “Deploying the application” on page 20.

The steps in this section are performed typically by the application deployer. To deploy the ITSO Bank application, do the following steps:

1. Create the DB2® database for ITSO Bank. This step is required.
2. Create an environment variable for ITSO Bank server. This step is optional.
3. Create an application server to host the application. This step is optional.
4. Customize the IBM® HTTP Server configuration. This step is optional.
5. Define a JDBC™ provider, data source, and authentication alias. This step is required if you are not using an Enhanced EAR.
6. Define virtual hosts. This step is optional and not required if you are using an Enhanced EAR.

If the application to be deployed is a WebSphere Enhanced EAR file, the resources configured in the Enhanced EAR file are created automatically when the application is deployed.

Creating the ITSO Bank DB2 database

To set up the DB2 database, make sure you have DB2 installed and running. Then run the following commands:

- ▶ Open a command prompt.
- ▶ Change directory to the `database\db2` folder in the `7672code` folder created when unzipping the additional material.
- ▶ Execute the `createbank.bat` file to define the database and table.
- ▶ Execute the `loadbank.bat` file to delete the existing data and add records.
- ▶ Execute the `listbank.bat` file to list the contents of the database.

Each command opens a new window where the DB2 script executes. Each command also leaves a connection to the database open, so you might want to execute a `db2 connect reset` command in each window opened to disconnect from the database so no unused connections are kept open.

Creating an environment variable

We recommend that you use WebSphere environment variables, rather than hard-coded paths when deploying an application. In the following sections, we assume that you have declared an `ITSOBANK_ROOT` variable. You will use it when specifying, for example, the JVM™ log's location.

Be certain you declare this variable at the right scope. For example, if you define this variable at the application server scope, it will only be known at that level. As long as you work with the WebSphere Application Server Base or Express editions, this is fine. But if you later decide to use the Network Deployment edition and you create a cluster of application servers, the `ITSOBANK_ROOT` variable will need to be defined at the cluster or cell level.

Use the steps outlined in Chapter 5, *Administration consoles and commands* to create a `ITSOBANK_ROOT` variable with a value of `C:\apps\ITSOBANK`.

There are several ways to organize WebSphere applications. Some companies prefer to create a directory for each application, as we do in our example, such as `C:\apps\application_name`, and keep all resources and directories required by the application in subdirectories under this directory. This strategy works well when deploying only one application per application server, again as we do in

our example, because the application server's log files could then all be changed to point to `c:\apps\application_name\logs`.

Other companies prefer to organize resources by resource type, and so create directories such as `c:\apps\logs\application_name.log`, `c:\apps\properties\application_name.properties`, and so on.

And some companies prefer to stick with the vendor defaults as far as possible. For WebSphere, that means that the applications are installed in the `profile_root/installedApps` directory and the logs files are written to the `profile_root/logs/server_name` directory.

Which option you choose is a matter of personal preferences and corporate guidelines.

Note: Make sure you create the target directory you specify for the `ITSOBANK_ROOT` variable before proceeding. If the directory is not created, the application server will not start.

Creating the ITSO Bank application server

In a distributed server environment, you have the option of using a single application server, or creating multiple application servers or clusters.

The advantages of deploying multiple applications to a single application server is that it consumes less resources. There is no overhead for any extra application server processes. Another benefit is that applications can make in-process calls to each other. For example, servlets in one EAR file could access Local interfaces of EJBs in another EAR file.

One alternative to using a single application server is to deploy each application to its own server. The advantages of deploying only one application on an application server is that it gives you greater control over the environment. The JVM heap sizes and environment variables are set at application server level, so all applications running in an application server share the JVM memory given to the application server and they would all see the same environment variables. Running each application in its own application server could also make it easier to perform problem determination. For example, if an application runs amok and consumes a lot of CPU, you could see which application it is by looking at the process ID of the application server.

In our example, we create a unique application server on which to run the ITSO Bank sample application.

Note: For a discussion of application server properties, see Chapter 6, *Administration of WebSphere processes*.

To create an application server, do the following steps:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Click the **New** button and provide the information node and server name, as shown in Figure 1.

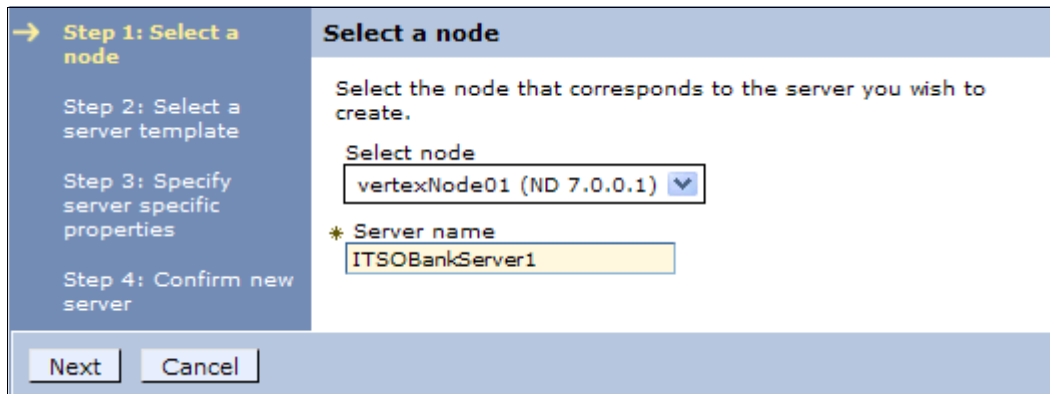


Figure 1 Creating the ITSO Bank application server

As you can see in the figure, the application server name will be ITSOBankServer1.

Click **Next**.

3. In Step 2, select which server template to use as the base for this new application server.

The DeveloperServer template is used when setting up a server for development use and will cause the JVM to prioritize for a quick start-up (by disabling bytecode verification, and performing JIT compilations with a lower optimization level). This option should not be used on a production server, where long run throughput is more important than early server startup.

If you have not created any templates on your own, then select the WebSphere **default**. Otherwise, select the server template you want to use and click **Next**.

4. In step 3, you can select to have WebSphere generate a unique set of port numbers for this application server. This ensures the ports defined for this server do not conflict with another server currently configured on this node. Check the **Generate Unique Http Ports** box and click **Next**.
5. On the Summary page, click **Finish**.

Changing the working directory

The next thing we want to do is to change the working directory for the application server process. This directory is the relative root for searching files. For example, if you do a `File.open("foo.gif")`, `foo.gif` must be present in the working directory. This directory will be created by WebSphere if it does not exist. We recommend that you create a specific working directory for each application server.

1. Select the server, **ITSOBankServer1**, you just created.
2. Expand the **Java™ and Process Management** in the Server Infrastructure section and select **Process Definition**.
3. Scroll down the page and change the working directory from `${USER_INSTALL_ROOT}` to `${ITSOBANK_ROOT}/workingDir`.
4. Click **OK**.

Note: The working directory will not be created automatically if you use a composed path, such as `C:/apps/ITSOBANK/workingDir`. If you want to use such a path, create it before starting the application server, or the startup sequence fails.

Changing the logging and tracing options

Next, we want to customize the logging and tracing properties for the new application server. There are several ways to access the logging and tracing properties for an application server.

- ▶ Select **Troubleshooting** → **Logs and Trace** in the navigation bar, then select a server.
- ▶ Select **Servers** → **Server Types** → **WebSphere application servers**, select a server, and then select **Logging and Tracing** from the Troubleshooting section.
- ▶ Select **Servers** → **Server Types** → **WebSphere application servers**, select a server, select **Process definition** from the Java and Process Management section. Select **Logging and Tracing** from the Additional Properties section.

Because we have just finished updating the application server process definition, we will take the third navigation path to customize the location of the JVM logs, the diagnostic trace logs, and the process logs.

1. Select **Logging and Tracing**.
2. Select **JVM Logs**.

This allows you to change the JVM standard output and error file properties. Both are rotating files. You can choose to save the current file and create a

new one, either when it reaches a certain size, or at a specific moment during the day. You can also choose to disable the output of calls to `System.out.print()` or `System.err.print()`.

We recommend that you specify a new file name, using an environment variable to specify it, such as:

```
${ITSOBANK_ROOT}/logs/SystemOut.log  
${ITSOBANK_ROOT}/logs/SystemErr.log
```

On this page you can also modify how WebSphere will rotate your log files.

Click **OK**.

3. Select **Diagnostic Trace**.

Each component of the WebSphere Application Server is enabled for tracing with the JRes interface. This trace can be changed dynamically while the process is running using the Runtime tab, or added to the application server definition from the Configuration tab. As shown in Figure 2, the trace output can be either directed to memory or to a rotating trace file.

Change the trace output file name so the trace is stored in a specific location for the server using the `ITSOBANK_ROOT` variable and select the **Log Analyzer** format.

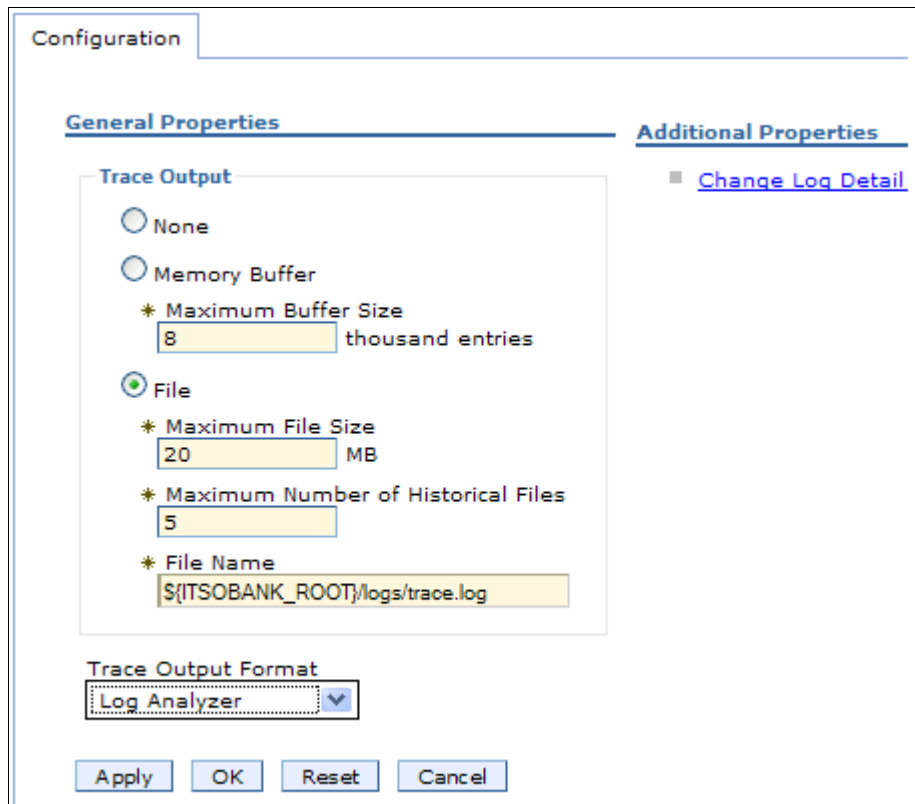


Figure 2 Specifying diagnostic trace service options

Click **OK**.

4. Select **Process Logs**.

Messages written by native code (JNI™) to standard out and standard error streams are redirected by WebSphere to process logs, usually called `native_stdout.log` and `native_stderr.log`. Change the native process logs to:

```

${ITSOBANK_ROOT}/logs/native_stdout.log
${ITSOBANK_ROOT}/logs/native_stderr.log

```

Click **OK**.

5. All log files produced by the application server are now redirected to the `${ITSOBANK_ROOT}/logs` directory. Save the configuration.

Note: The rest of this example assumes a default HTTP port of 9080 for the Web container. Before proceeding, check the application server you created to determine the port you should use:

1. Select **Servers** → **Server Types** → **WebSphere application servers**.
2. Select the **ITSOBankServer1**.
3. Select **Ports** in the Communications section.
4. Scroll down the page and note the port listed for WC_defaulthost.

Defining the ITSO Bank virtual host

Enhanced EAR file users: If you are using an Enhanced EAR file, the virtual host can be defined at packaging time. See Chapter 14, *Packaging applications for deployment*.

Web modules need to be bound to a specific virtual host. For our sample, we chose to bind the RAD75EJBWeb Web module to a specific virtual host called `itsobank_host`. This virtual host has the following host aliases:

- ▶ `www.itsobank.ibm.com:80`
- ▶ `www.itsobank.ibm.com:9080`

Any request starting with `itsobank_host_alias/RAD75EJBWeb`, such as `http://www.itsobank.ibm.com:9080/RAD75EJBWeb`, is served by the RAD75EJBWeb application.

Tip: You can restrict the list of hosts used to access the Web application by removing hosts from the virtual host definition.

Imagine you want to prevent users from directly accessing the ITSO Bank application from the WebSphere internal HTTP server when they invoke `http://www.itsobank.ibm.com:9080/RAD75EJBWeb`. In other words, you want to force all requests to go through the Web server plug-in. You can achieve this by removing `www.itsobank.ibm.com:9080` from the virtual host aliases list.

To create the `itsobank_host` virtual host, do the following steps:

1. Select the **Environment** → **Virtual Hosts** entry in the navigation pane.
2. Click **New**.
3. Enter the virtual host name, `itsobank_host`.
4. Click **Apply**.
5. Select **Host Aliases** in the Additional Properties section.

6. Add the two aliases shown in Figure 3 by clicking **New**, entering the values, and clicking **OK**.



Figure 3 WebSphere Bank virtual host aliases

7. Click **OK**.
8. Save the configuration.

Creating the virtual host for IBM HTTP Server and Apache

Now that we have defined a `itsobank_host` virtual host, we need to configure the Web server to serve the host aliases in the virtual host. The steps below are valid for both the IBM HTTP Server V7 and Apache 2.x.

Configuring virtual hosting

Note: It is not necessary to create a virtual host in `httpd.conf`. It is required only if you want to customize the configuration, for example, by separating the logs for each virtual host. This is not normally done.

Creating virtual hosts is done using the `VirtualHost` directive, as in Example 1.

Example 1 Using VirtualHost

```
<VirtualHost www.itsobank.ibm.com:80>
  ServerAdmin webmaster@itsobank.ibm.com
  ServerName www.itsobank.ibm.com
  DocumentRoot "C:\IBM\HTTPServer\htdocs\itsobank"
  ErrorLog logs/itsobank_error.log
  TransferLog logs/itsobank_access.log
</VirtualHost>
```

If you want to have multiple virtual hosts for the same IP address, you must use the `NameVirtualHost` directive. See Example 2.

Example 2 Using the `NameVirtualHost` and `VirtualHost` directives

NameVirtualHost 9.11.12.13:80

```
<VirtualHost itso_server:80>
  ServerAdmin webmaster@itso_server.com
  ServerName itso_server
  DocumentRoot "C:\IBM\HTTPServer\htdocs\itso_server"
  ErrorLog logs/itso_server_error.log
  TransferLog logs/itso_server_access.log
</VirtualHost>

<VirtualHost www.itsobank.ibm.com:80>
  ServerAdmin webmaster@itsobank.ibm.com
  ServerName www.itsobank.ibm.com
  DocumentRoot "C:\IBM\HTTPServer\htdocs\itsobank"
  ErrorLog logs/itsobank_error.log
  TransferLog logs/itsobank_access.log
</VirtualHost>
```

The `www.itsobank.ibm.com` and the `itso_server` hosts have the same IP address, 9.11.12.13. We have set this by inserting the following line in the machine hosts file, located in `%windir%\system32\drivers\etc` or in `/etc` on UNIX® systems):

```
9.11.12.13 www.itsobank.ibm.com itso_server
```

In a real-life environment, this would probably be achieved by creating aliases at the DNS level. In any event, you must be able to ping the host you have defined, using commands such as `ping www.itsobank.ibm.com`.

As you can see in Example 2, each virtual host has a different document root. Make sure that the directory you specify exists before you start the HTTP server. While testing the setup, you can place an `index.html` file at the document root stating which virtual host is being called. This lets you easily see which virtual host is being used.

You must restart the IBM HTTP Server to apply these changes. If you are running a Windows® system, we recommend that you try to start the server by running `apache.exe` from the command line rather than from the Services window. This allows you to spot error messages thrown at server startup.

If your virtual hosts are correctly configured, invoking `http://www.itsobank.ibm.com` or `http://itso_server` returns different HTML pages.

Creating a DB2 JDBC provider and data source

Enhanced EAR file users: If you are using an Enhanced EAR file, the JDBC provider, data source, and J2C authentication entry can be defined at packaging time. See Chapter 14, *Packaging applications for deployment*.

The ITSO Bank sample application uses a relational database, via Java Persistence API, to store information. To access this database, a data source needs to be defined with a JNDI name that matches the data source configuration in the JPA module's `persistence.xml` file. The ITSO Bank sample application is configured for Derby by default. In Chapter 14, *Packaging applications for deployment*, however, we modified the ITSO Bank application to run against a DB2 database instead. We will now create the DB2 JDBC provider, data source, and JAAS authentication alias required to run against DB2.

For detailed information about JDBC providers and data sources, refer to Chapter 9, *Accessing databases from WebSphere*.

Configuring environment variables for DB2 JDBC driver

For the DB2 JCC JDBC Provider to find its classes, the `DB2_JCC_DRIVER_PATH` and `DB2_JCC_DRIVER_NATIVEPATH` environment variables must be set up. To set up these variables, do the following steps:

1. Select **Environment** → **WebSphere Variables**.
2. Locate and click the **DB2_JCC_DRIVER_PATH** entry.
3. In the value field, enter the path to where the DB2 JDBC driver is located. For example, for DB2, the location is likely to be:

`C:\Program Files\IBM\SQLLIB\java`

See Figure 4.

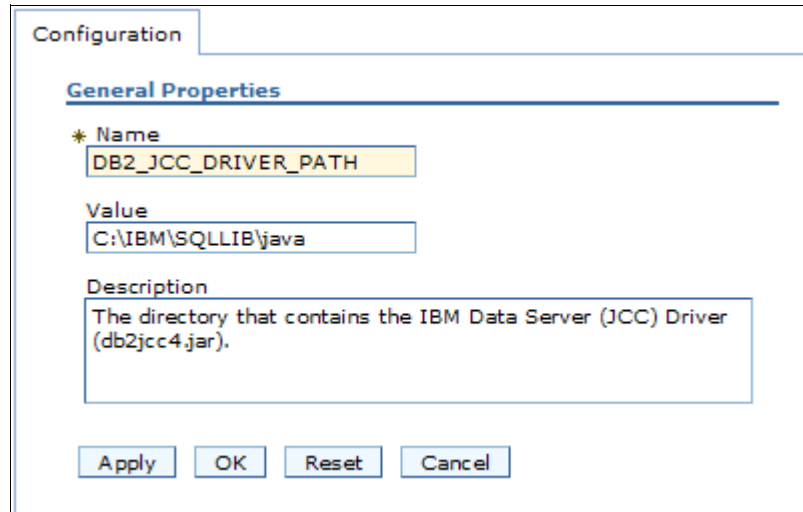


Figure 4 Configuring DB2 Driver Path

Click **OK**.

4. Repeat the process for the DB2_JCC_DRIVER_NATIVEPATH variable. For DB2, it should use the same path, C:\Program Files\IBM\SQLLIB\java.

Configuring J2C authentication data

The user ID and password required to access the database are specified in a J2C authentication data entry:

1. Select **Security** → **Global Security**. Expand the Java Authentication and Authorization Service section under the Authentication section and select **J2C authentication data**.

2. Click **New**, and specify the following information to create the authentication data. Once completed, the authentication information should be similar to Figure 5.



The screenshot shows a dialog box titled "Global security" with a breadcrumb path: "Global security > JAAS - J2C authentication data > New". Below the breadcrumb, there is a descriptive text: "Specifies a list of user identities and passwords for Java(TM) 2 connector security to use." The dialog is divided into a "General Properties" section. It contains four fields: "Alias" with the value "itsobank", "User ID" with the value "db2inst1", "Password" with a masked value of "*****", and "Description" with the value "ITSO Bank authentication alias". At the bottom of the dialog, there are four buttons: "Apply", "OK", "Reset", and "Cancel".

Figure 5 Creating ITSO Bank JAAS authentication alias

3. Click **OK**.

Creating the ITSO Bank JDBC provider

The following steps take you through the creation of a JDBC provider targeting a DB2 database. To create a JDBC provider from the administrative console, do the following steps:

1. Expand the **Resources** entry and then the **JDBC** entry. Then select the **JDBC Providers** entry.
2. Select the scope of this resource. In a stand-alone server environment, it is sufficient to create the data source at the server level. Otherwise, define it at the cluster or cell level. A rationale for this is to be able to share the definition across multiple servers in a cluster. To change this, select the server you are deploying to in the scopes list.
3. Click the **New** button.
4. In the Configuration dialog box, select the general properties for the JDBC provider, as shown in Figure 6.

→ **Step 1: Create new JDBC provider**

Step 2: Enter database class path information

Step 3: Summary

Create new JDBC provider

Set the basic configuration values of a JDBC provider, which encapsulates the specific vendor JDBC driver implementation classes that are required to access the database. The wizard fills in the name and the description fields, but you can type different values.

Scope

* Database type

* Provider type

* Implementation type

* Name

Description

Figure 6 Creating a DB2 JDBC provider

- Database type: DB2
- Provider type: DB2 Using IBM JCC Driver
- Implementation type: XA data source
- Name: DB2 Using IBM JCC Driver (XA)

Click **Next**.

Note: We used the DB2 XA-capable JDBC Driver for the ITSO Bank sample. If your application does not require two-phase commit capabilities, use the regular driver. If using an XA-capable driver, it is a best practice to indicate that it is an XA-capable driver by including XA in its name, such as MyJDBCdriverXA.

5. The next window allows you to change the location for the JDBC driver files, but because we configured the paths earlier, we do not need to do it again. Click **Next**.
6. On the Summary page, click **Finish**.

Creating the ITSO Bank data source

The next step is to create the data source for the ITSO Bank DB2 database. To create a data source, do the following steps:

1. Select **Resources** → **JDBC** → **JDBC Providers**.
2. Select the **DB2 Using IBM JCC Driver (XA)** and select **Data Sources** under Additional Properties.
3. Click **New** to add the new data source. See Figure 7.

Create a data source

Create a data source

→ **Step 1: Enter basic data source information**

Step 2: Enter database specific properties for the data source

Step 3: Setup security aliases

Step 4: Summary

Enter basic data source information

Set the basic configuration values of a datasource for association with your JDBC provider. A datasource supplies the physical connections between the application server and the database.

Requirement: Use the Datasources (WebSphere(R) Application Server V4) console pages if your applications are based on the Enterprise JavaBeans(TM) (EJB) 1.0 specification or the Java(TM) Servlet 2.2 specification.

Scope

JDBC provider name

* Data source name

* JNDI name

Figure 7 ITSO Bank basic data source properties

- Data source name
 Enter the data source name, which must be unique in the administrative domain or cell. We recommend that you use a value indicating the name of the database this data source is targeting, such as “ITSOBankDS”.
 - JNDI name
 Enter the name by which applications access this data source. If not specified, the JNDI name defaults to the data source name prefixed with jdbc/. For the ITSO Bank, set this field to jdbc/itsobank. This value can be changed at any time after the data source has been created.
4. Click **Next**. On the second page, enter the information shown in Figure 8.

Step 1: Enter basic data source information

→ **Step 2: Enter database specific properties for the data source**

Step 3: Setup security aliases

Step 4: Summary

Enter database specific properties for the data source

Set these database-specific properties, which are required by the database vendor JDBC driver to support the connections that are managed through the datasource.

Name	Value
* Driver type	<input style="width: 80%;" type="text" value="2"/>
* Database name	<input style="width: 80%;" type="text" value="ITSOBANK"/>
* Server name	<input style="width: 80%;" type="text"/>
* Port number	<input style="width: 80%;" type="text" value="50000"/>

Use this data source in container managed persistence (CMP)

Previous
Next
Cancel

Figure 8 ITSO Bank data source database properties

- Driver type

Select the driver type to use.

If the database is on the same machine as the WebSphere installation you can use a type 2 driver and do not need to enter a server name. If the database is on a remote machine you use either a type 4 driver, which allows WebSphere to connect remotely to the database over TCP/IP, or a type 2 driver.

To use a type 2 driver with a remote database you need a local DB2 Client installation on the WebSphere machine and catalog the database on the DB2 Client. WebSphere then sees the database as local and the DB2 Client handles the remote calls. In our setup the database is on the same machine as the WebSphere installation so we choose a type 2 driver and do not enter a Server name.

- Database name

Enter the name of the database, ITSOBANK in our example.

- Port number

Our DB2 installation uses port 50000 so we keep the default value.

- Use this Data Source in container-managed persistence (CMP)

Because the ITSO Bank uses Java Persistence API for database persistence instead of CMP EJBs the data source does not need to be set up for CMP EJBs. So uncheck this option.

5. Click **Next**. On the third page, enter the information shown in Figure 9.

The screenshot shows a wizard titled "Setup security aliases". On the left, a sidebar lists four steps: "Step 1: Enter basic data source information", "Step 2: Enter database specific properties for the data source", "Step 3: Setup security aliases" (highlighted with a yellow arrow), and "Step 4: Summary". The main content area has a blue header "Setup security aliases" and a sub-header "Select the authentication values for this resource." Below this are four dropdown menus: "Authentication alias for XA recovery" (set to "(none)"), "Component-managed authentication alias" (set to "(none)"), "Mapping-configuration alias" (set to "(none)"), and "Container-managed authentication alias" (set to "vertexCellManager/itsobank"). A note at the bottom states: "Note: You can create a new J2C authentication alias by accessing one of the following links. Clicking on a link will cancel the wizard and your current wizard selections will be lost." Below the note are two blue links: "Global J2C authentication alias" and "Security domains". At the bottom of the wizard are three buttons: "Previous", "Next", and "Cancel".

Figure 9 ITSO Bank database security alias properties

- Container-managed authentication alias

This is the preferred method for specifying authentication information for the database. Enter the J2C alias used for connecting to the data source by selecting the authentication alias created previously, *cell name/itsobank*.

6. Click **Next**, and then on the summary page, click **OK**.
7. Save the configuration.
8. Test the connection by selecting the data source and clicking the **Test Connection** button.

Deploying the application

In this section, we show the steps required to deploy the application to WebSphere Application Server. The EAR file we deploy is the RAD75EJBWebEAR file, which is a regular EAR file and not an Enhanced EAR file.

Follow these steps to deploy the application:

1. Select **Applications** → **New Application** from the administrative console navigation bar and click the **New Enterprise Application** on the panel shown.

Note: In Chapter 14, *Packaging applications for deployment*, we created a business level application and added this application to it. As a result, the application was installed. With the exception of this first step, the rest of this process reflects the steps you will see regardless of whether you are installing the application as the result of adding as an asset to a BLA, or are installing the application and creating a BLA as part of the process.

2. Check the **Local file system** box and click the **Browse** button to locate the RAD75EJBWebEAR.ear file. Select the file and click **Open**.

From the install windows, you can install files that are located either on the same machine as the browser you are using to access the WebSphere administrative console, the local file system option, or on the WebSphere Application Server itself, the remote file system option. If you select the Local file system option, the administrative console automatically uploads the file you select to the application server, or to the deployment manager if this is a distributed server environment. If you select the Remote file system check box, you can browse all the nodes in the cell to find the file. The file is then, if necessary, uploaded to the application server or deployment manager.

When you have made your selection, click the **Next** button.

3. WebSphere Application Server allows you to take a shortcut when installing an application. If you select the **Fast Path - Prompt only when additional information is required** option, only the windows where you actually need to fill out some information during installation are shown.

For this example, however, we will explain the options, so select **Detailed - Show all installation options and parameters**.

If you expand the **Choose to generate default bindings and mapping** you can alter the bindings for the application you are deploying. If you select the **Generate Default Bindings** option, WebSphere Application Server completes any incomplete bindings in the application with default values, but

it does not alter any existing bindings. Checking the **Override existing bindings** allows you to specify a bindings file which contains new bindings.

The contents of the application or module that you are installing determines which options are displayed on the bindings page. For our ITSO Bank application the options documented in Table 1 are displayed.

Because our application uses JEE 5 and EJB 3.0 and rely on the bindings and mappings generated automatically by the EJB container there is no need to override our bindings, so we leave all check boxes cleared.

Table 1 Application default bindings

Binding name	Detailed information
Specific bindings file	You can create a specific bindings file using your favorite editor and load it during application installation by clicking Browse next to the specific bindings file.
Unique prefix for beans	You can generate default EJB JNDI names using a common prefix. EJBs for which you did not specify a JNDI name will get a default name, built by concatenating the prefix and the EJB name. If you specify a prefix of myApp/ejb, then JNDI names default to myApp/ejb/EJBName, such as myApp/ejb/Account.
Virtual host bindings	You can bind all Web modules to a specific virtual host, such as itsobank_host.

Click **Next**.

4. The rest of the wizard is divided into steps. The number of steps depends on your application, for example, if it contains EJB modules or Web modules, you will see windows prompting for the information necessary to deploy them.
5. Step 1: Select installation options.

Step 1 gives you a chance to review the installation options. You can specify various deployment options, such as JSP™ precompiling, and whether you want to generate EJB deployment code (not applicable for EJB 3.0 beans).

- If you are deploying an Enhanced EAR file, this is where you make the decision whether to use the resource configuration information packaged in the Enhanced EAR file or not. If the EAR file you are installing is an Enhanced EAR, the install window preselects the **Process embedded configuration** check box. If you do not want to use the resource configuration information packaged in the Enhanced EAR file, you must deselect this check box. Because we have already configured the necessary resources needed, we make sure the **Process embedded configuration** check box is not selected.
- Selecting the **Pre-compile JavaServer™ Pages files** option makes WebSphere compile all JSPs in the EAR file during install time. This

causes the time-consuming task of JSP compilation to be performed during install time instead of during runtime, preventing the first user that accesses the application to pay that penalty.

A second alternative to pre-compiling JSPs is to use the JspBatchCompiler script found in the bin directory of the profile you are using, to compile the JSPs after the application has been installed.

- This page also allows you to specify file permissions for files in your application. To use one of the predefined file permissions, select it, and then click **Set file permissions**. You can also specify your own file permissions using regular expressions.
- The administrative console displays the Application Build ID of the application being installed. This string is specified in the MANIFEST.MF file in the EAR file's META-INF folder and can be set using the Rational Application Developer Assembly and Deploy tool.

The following is an example of a version number specified in the MANIFEST.MF file:

```
Implementation-Version: Version 1.2.3
```

- The dispatching and servicing of remote resources are extensions to the Web container that allows frameworks, servlets, and JSPs to include content from outside of the current executing resource's JVM as part of the response sent to the client.


To enable these features, select the corresponding check boxes to allow dispatching or servicing includes to/from remote resources.

- The **Allow EJB reference targets to resolve automatically** option is used for EJB 2.1 or earlier or Web 2.3 or earlier modules and allows WebSphere Application Server to provide a default value or automatically resolve EJB references for any EJB reference that does not have a binding. Because our application is at EJB 3.0 this option does not apply to our application.

Click **Next**.

6. Step 2: Map modules to servers.

Select the server on which you want each module deployed. For better performance, we recommend that you deploy all modules from one application in a single server. Especially, do not separate the EJB clients, usually servlets in Web modules, from the EJBs themselves.

Click the  icon to select all modules in the ITSO Bank EAR file. In the Clusters and Servers box, select **ITSOBankServer1**. Then click **Apply**. This assigns all modules to the ITSOBankServer1 application server. If you deploy to a cluster, select the cluster instead of the single application server.

See Figure 10.

Web servers: If you have a Web server defined, select both the Web server and ITSOBankServer1 in the server list. Press and hold the CTRL key to select multiple servers. Mapping Web modules to Web servers ensures the Web server plug-in will be generated properly.

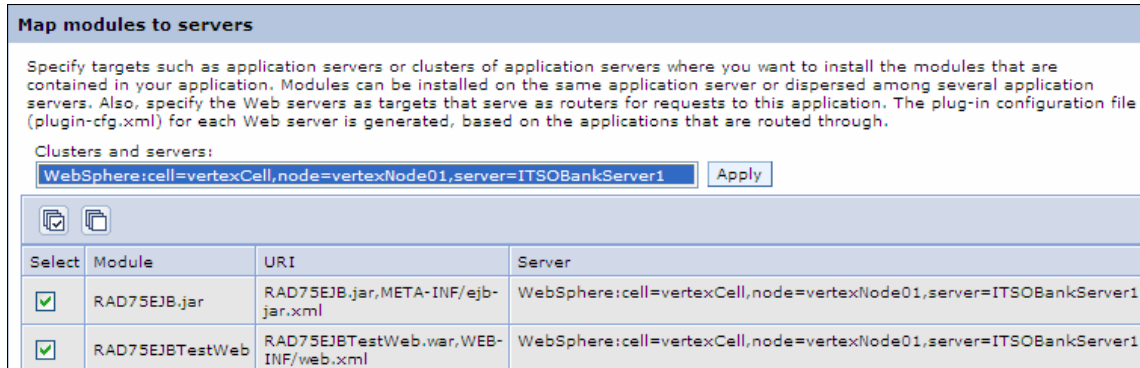


Figure 10 Mapping modules to application servers

Click **Next**.

7. Step 3: Provide JSP reloading options for Web modules.

This setting allows you to configure if and how often WebSphere should check for updates to JSP files, and if they should be reloaded or not. In a production environment, you might want to disable this to improve performance.

Click **Next**.

8. Step 4 and 5: Map shared libraries, and Map shared library relationships

If your application depends on shared libraries, you can specify them here. Click **Next**.

9. Step 6: Initialize parameters for servlets.

For servlets that honor initialization parameters (specified by the init-param tag in the Web module's web.xml deployment descriptor) you can configure the value of the parameters.

10. Step 7: Provide JNDI names for beans.

Use this window to bind the enterprise beans in your application or module to a JNDI name. Because our application is at EJB 3.0 level we can leave it blank to have WebSphere Application Server use the default names.

Click **Next**.

11. Step 8: Bind EJB Business interfaces to JNDI names.

This window allows you to specify a JNDI name for the business interfaces of your EJBs. Because our application is at EJB 3.0 level we can leave it blank to have WebSphere Application Server assign default JNDI name.

12. Step 9: Map EJB References to beans.

Each EJB reference defined in your application must be mapped to an enterprise bean. Because our Web module is at Web 2.5 level we can leave it blank to have WebSphere Application Server use the default names. If the reference was in an EJB 2.x or earlier or Web 2.3 or earlier module we could check the **Allow EJB reference targets to resolve automatically** and would then not need to specify a target JNDI name either.

Click **Next**.

13. Step 10: Map virtual hosts for Web modules.

Select the virtual host we created for the application (itsobank_host).

Note: If deploying an Enhanced EAR file with a virtual host configured the install panel does not display and make the virtual host name selectable. Instead only those virtual host definitions configured in the WebSphere environment are displayed. To map the Web modules to the virtual host defined in the Enhanced EAR file you need to configure that after deploying the application.

Click **Next**.

14. Step 11: Map context roots for Web modules.

Select the context root to bind the module against.

Click **Next**.

15. Step 12: Metadata for modules.

Checking the metadata-complete attribute tells WebSphere Application Server to ignore any deployment information specified in source code annotations. Leave both check boxes cleared to use the information from the annotations.

Click **Next**.

16. Step 13: Summary.

The Summary window gives an overview of application deployment settings. If those settings are fine, click **Finish** to deploy the application.

17. Save the configuration.

If you are working in a distributed server environment, make sure you synchronize the changes with the nodes so the application is propagated to the target application servers.

18. If you mapped the Web modules to a Web server, make sure the Web server plug-in is regenerated and propagated to the Web server. For a quick refresh, restart the Web server.

Deployment of the RAD75EJBWebEAR application is now complete. After starting the application you can now verify that the application works by pointing your browser to:

`http://www.itsobank.ibm.com:9080/RAD75EJBWeb`

If successful it should display the Web page for the ITSO Bank application. Click the **RedBank** button and provide customer number 222-22-2222 to see an example of a customer's accounts, as shown in Figure 11.

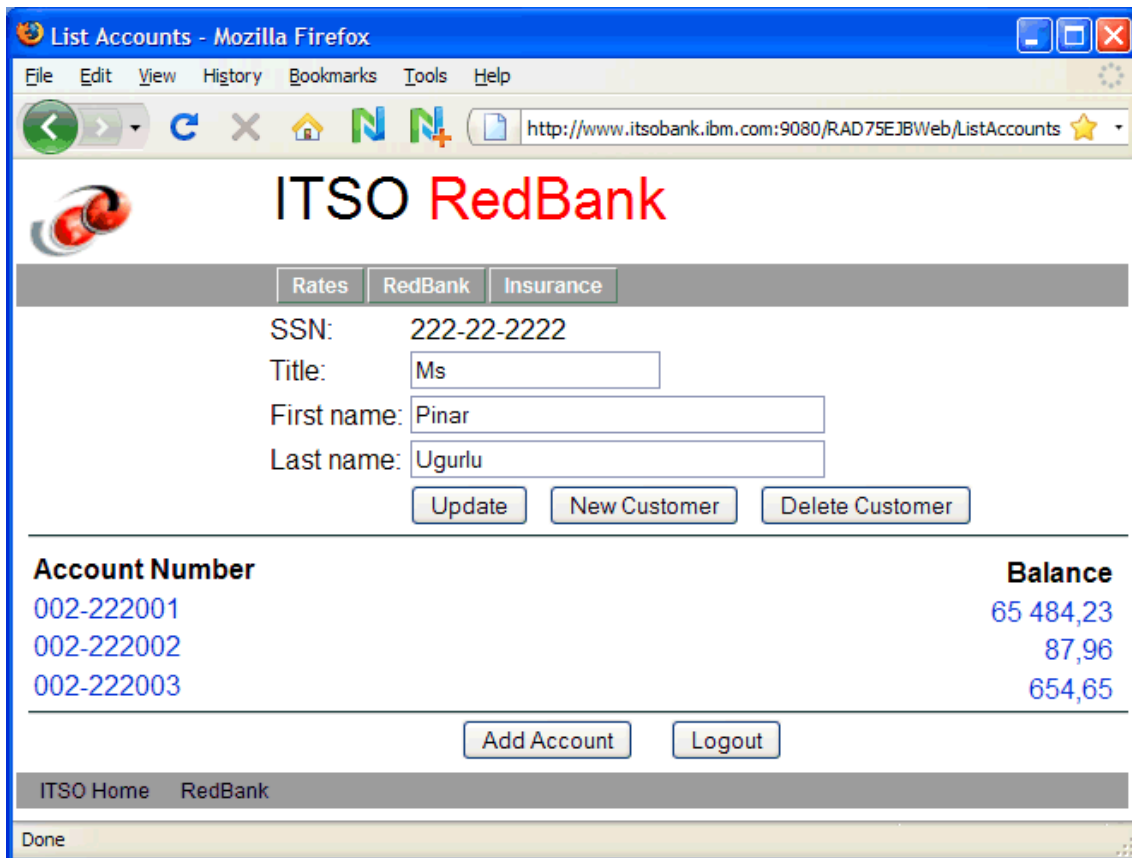


Figure 11 ITSO Bank Web application

If you have any problems related to virtual hosts, restart the server and try again. WebSphere Application Server might need a restart to pick up virtual hosts changes.

Deploying application clients

To run a Java-based client/server application, the client application executes in a client container of some kind. You might, for example, use a graphical Swing application that calls EJBs on an application server. WebSphere Application Server V7 supports several different application client environments:

▶ **Java EE client (JEE client):**

This client uses services provided by the JEE client container.

This client is a Java application program that accesses EJBs, JDBC databases, and JMS queues. The JEE application client program runs on client machines. This program allows the same Java programming model as other Java programs. However, the JEE application client depends on the application client runtime to configure its execution environment, and it uses the JNDI name space to access resources, the same as you would in a normal server application (like a servlet).

The JEE application client brings the JEE programming model to the client, and provides:

- XML deployment descriptors
- JEE naming (`java:comp/env`), including EJB references and resource references

The JEE application client is launched using the `launchClient` script, which sets up the environment with the necessary classpaths, and so on, for you.

▶ **Java thin client:**

This client does not use services provided by the JEE client container.

This client provides a lightweight Java client programming model and is best suited for use in situations where a Java client application exists, but the application must be enhanced to make use of EJBs. It can also be used where the client application requires a thinner, more lightweight environment than the one offered by the JEE application client. The thin client supports JVMs from IBM, Sun™ and HP-UX. When launching the thin application client, you must set up the correct classpaths yourself and make sure that the required libraries for your application and the WebSphere libraries are included.

▶ Pluggable application client:

This client does not use services provided by the JEE Client Container.

This client is similar to the Thin application client, but does not include a JVM. The user is required to provide a JVM, and it can use the Sun JDK™ or the IBM JDK.

Note: The pluggable client is deprecated in WebSphere Application Server V7, and replaced by the Java thin client.

▶ Applet application client:

In the Applet client model, a Java applet embedded in an HTML document executes in a Web browser. With this type of client, the user accesses an enterprise bean in the application server through the Java applet in the HTML document.

▶ ActiveX® to EJB Bridge application client:

The ActiveX application client allows ActiveX programs to access enterprise beans through a set of ActiveX automation objects. The ActiveX application client uses the Java Native Interface (JNI) architecture to programmatically access the Java virtual machine (JVM) API. Therefore, the JVM code exists in the same process space as the ActiveX application (Visual Basic®, VBScript, or Active Server Pages files) and remains attached to the process until that process terminates. The ActiveX to EJB Bridge is supported on Windows only.

For detailed capabilities of each client container, search the Information Center for Client Applications, or visit:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.webSphere.nd.multipatform.doc/info/ae/ae/cccli_clientapps.html

Install the application client environments from the WebSphere installation windows by selecting the **Launch the installation wizard for WebSphere Application Clients** option. The installation package contains the following installable components:

- ▶ IBM Java Runtime Environment (JRE™), or an optional full Software Development Kit
- ▶ Java EE application client and Java thin application client
- ▶ ActiveX to EJB Bridge runtime for ActiveX to EJB Bridge application client applications (only for Windows)
- ▶ Pluggable Client (deprecated)
- ▶ Samples for the various application client containers

Note: The JEE client is automatically installed as part of a full WebSphere install. In other words, if you will run the client application on a machine that already has WebSphere installed, you do not need to install the WebSphere JEE client on top.

Defining application client bindings

The ITSO Bank application also has a standalone application client which we will use to demo the WebSphere Application Server application client container.

First we need to import the project containing the standalone application client into Rational Application Developer Assembly and Deploy and export it as an EAR file. Refer to Chapter 14, *Packaging applications for deployment* for details on how to perform such tasks. Do the following steps:

1. Import the appclient\RAD75AppClient.zip Project Interchange™ file from the 7672codesolution.zip file. Select both projects when importing.
2. Because the RAD75AppClient project has a specific binding configured for the EJBBankBean in the RAD75EJBEBEAR file we need to modify this binding to point to the same bean in the RAD75EJBWebEAR file, which is the application we have installed on our server.
 - a. To do this expand the **RAD75AppClient** project and double-click the **RAD75AppClient** deployment descriptor.
 - b. Then click the **Open WebSphere Bindings** link from the right panel.
 - c. Select the **EJB Reference (ejb/bank)** and change the name from `ejb/RAD75EJBEBEAR/RAD75EJB.jar/EJBBankBean#itso.bank.service.EJBBankRemote` to `itso.bank.service.EJBBankRemote`.

By using the short name we rely on the Autolink feature to search for and invoke a matching EJB interface. This works fine in our environment where we only have one instance of the EJB installed and running.
 - d. Press **Ctrl-S** to save the deployment descriptor.
3. Export the **RAD75AppClientEAR** project and its RAD75AppClient module as described in Chapter 14, *Packaging applications for deployment*.

Launching the J2EE client

A JEE client application needs a container to run in. In this example, we will use the JEE application client container. This container can be started using the launchClient program in the `install_root/bin` directory. The launchClient program has the following syntax:

Usage: launchClient [-profileName pName | -JVMOptions options | -help |
-?] <userapp> [-CC<name>=<value>] [app args]

The elements of syntax are:

-profileName	This option defines the profile of the application server process in a multi-profile installation. The -profileName option is not required for running in a single profile environment or in an application client installation. The default is default_profile.
-JVMOptions	This is a valid Java standard or nonstandard option string. Insert quotation marks around the option string.
-help, -?	Print the usage information.
<userapp.ear>	Type the path/name of the .ear file containing the client application.

The -CC properties are for use by the application client runtime. There are numerous parameters available and because of this we only describe the more commonly used ones. For full explanation of all parameters, execute **launchClient -help**.

-CCverbose	Use this option with <true false> to display additional informational messages. The default is false.
-CCclasspath	This property is a classpath value. When an application is launched, the system classpath is not used. If you need to access classes that are not in the EAR file or part of the resource classpaths, specify the appropriate classpath here. Multiple paths can be concatenated.
-CCjar	This is the name of the client JAR file within the EAR file that contains the application you want to launch. This argument is only necessary when you have multiple client JAR files in the EAR file.
-CCBootstrapHost	This option is the name of the host server you want to connect to initially.
-CCBootstrapPort	This option is the server port number. If not specified, the WebSphere default value (2809) is used.
-CCproviderURL	This option provides bootstrap server information that the initial context factory can use to obtain an initial context. A WebSphere Application Server initial context factory can use either a CORBA object URL or an IIOP URL. CORBA object URLs are more flexible than IIOP URLs and are the recommended URL format to use. This value can contain more than one bootstrap server address. This feature can

be used when attempting to obtain an initial context from a server cluster. In the URL, you can specify bootstrap server addresses for all servers in the cluster. The operation will succeed if at least one of the servers is running, eliminating a single point of failure. The address list does not process in a particular order. For naming operations, this value overrides the `-CCBootstrapHost` and `-CCBootstrapPort` parameters. An example of a CORBA object URL specifying multiple systems is:
`-CCproviderURL=corbaloc:iiop:myserver.mycompany.com:9810,:mybackupserver.mycompany.com:2809`

- `-CCtrace` Use this option with `<true | false>` to have WebSphere write debug trace information to a file. The value `true` is equivalent to a trace string value of `com.*=all=enabled`. Instead of the value `true` you can specify a trace string, for example, `-CCtrace=com.ibm.ws.client.*=all=enabled`. Multiple trace strings can be specified by separating them with a colon (:). You might need this information when reporting a problem to IBM Service. The default is `false`.
- `-CCtracefile` This option is the name of the file to which to write trace information. The default is to output to the console.
- `-CCpropfile` This option is the name of a properties file containing `launchClient` properties. In the file, specify the properties without the `-CC` prefix. For example: `verbose=true`.

The app args are for use by the client application and are ignored by WebSphere.

To start the ITSO Bank standalone application client using the `launchClient` command, execute the command shown in Example 3.

Example 3 Launching ITSO Bank standalone application client

```
C:\IBM\WebSphere\AppServer\profiles\ITSOBank\bin>launchClient.bat  
c:\RAD75AppClientEAR_withModifiedBinding.ear
```

```
IBM WebSphere Application Server, Release 7.0  
Java EE Application Client Tool  
Copyright IBM Corp., 1997-2008  
WSCL0012I: Processing command line arguments.  
WSCL0013I: Initializing the Java EE Application Client Environment.  
[2009-04-04 13:42:04:546 CEST] 00000000 W UOW=null  
source=com.ibm.ws.ssl.config.SSLConfig org=IBM prod=WebSphere  
component=Application Server thread=[P=321234:  
O=0:CT]
```

```
CWPKI0041W: One or more key stores are using the default
password.
WSCLO035I: Initialization of the Java EE Application Client Environment
has completed.
WSCLO014I: Invoking the Application Client class
itso.rad75.client.control.BankDesktopController
```

The application will open and display a graphical window, as shown in Figure 12.

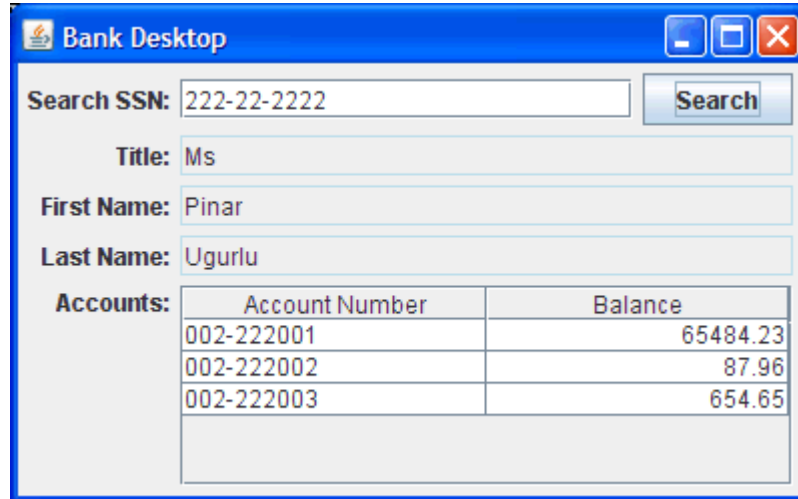


Figure 12 Running ITSO Bank standalone application client

Updating applications

WebSphere Application Server has features that allow applications to be updated and restarted at a fine-grained level. It is possible to update only parts of an application or module and only the necessary parts are restarted. You can:

- ▶ Replace an entire application (.ear file).
- ▶ Replace, add, or remove a single module (.war, EJB™ .jar, or connector .rar file).
- ▶ Replace, add, or remove a single file.
- ▶ Replace, add and remove multiple files by uploading a compressed file describing the actions to take.

If the application is running while being updated, WebSphere Application Server automatically stops the application, or only its affected components, updates the application, and restarts the application or components.

When updating an application, only the portion of the application code that changed needs to be presented to the system. The application management logic calculates the minimum actions that the system needs to execute in order to update the application. Under certain circumstances, the update can occur without stopping any portion of the running application.

WebSphere Application Server also has support for managing applications in a cluster for continuous availability. The action, Rollout Update, sequentially updates an application installed on multiple cluster members across a cluster. After you update an application's files or configuration, use the Rollout Update option to install the application's updated files or configuration on all cluster members of a cluster on which the application is installed.

Rollout Update does the following for each cluster member in sequence:

1. Saves the updated application configuration.
2. Stops all cluster members on a given node.
3. Updates the application on the node by synchronizing the configuration.
4. Restarts the stopped cluster members on that node.

This action updates an application on multiple cluster members while providing continuous availability of the application.

Replacing an entire application EAR file

To replace a full EAR with a newer version, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application update window, select the **Replace the entire application** option.
3. Select either the **Local file system** or **Remote file system** option. Click the **Browse** button to select the updated EAR file. Click **Next**.
4. Proceed through the remaining windows and make any changes necessary. For information about the windows, see “Deploying the application” on page 20. On the Summary window, click **Finish**.
5. When the application has been updated in the master repository, select the **Save** link.
6. If you are working in a distributed server environment, make sure that you also synchronize the changes with the nodes.

7. If the application update changes the set of URLs handled by the application (servlet mappings added, removed, or modified), make sure the Web server plug-in is regenerated and propagated to the Web server.

Note: It might take a few seconds for the WebSphere runtime to pick up the changes and restart the application as necessary. If your changes do not seem to have effect, wait and try again. You can also look at the SystemOut.log file for the application server to see when it has restarted the application.

Replacing or adding an application module

To replace only a module, such as an EJB or Web module of an application, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application update window, select the **Replace or add a single module** option.
3. In the Specify the path beginning with the installed application archive file... field, enter the relative path to the module to replace. For example, if you were to replace the HelloWeb module, enter HelloWeb. If you enter a path or file that does not exist in the EAR file, it will be added.
4. Select either the **Local file system** or **Remote file system** option and click the **Browse** button to select the updated module.
5. Click **Next**.
6. Proceed through the remaining windows and make any necessary changes. For information about the windows, see “Deploying the application” on page 20. On the Summary window, click **Finish**.

Note: If you are adding a Web module, make sure you select the detailed install option. This allows you to select the correct target server for the module in the Map modules to servers step, as well as specifying a context root for the module.

7. When the application has been updated in the master repository, select the **Save** link.
8. If you are working in a distributed server environment, make sure that you also synchronize the changes with the nodes.

9. If the application update changes the set of URLs handled by the application (servlet mappings added, removed, or modified), make sure the Web server plug-in is regenerated and propagated to the Web server.

Tip: Modules can also be managed using the Manage Modules page. Select **Applications** → **Application Types** → **WebSphere enterprise applications** and click the link for the application. Then click the **Manage Modules** link in the Modules section. Select the module to modify and then click the **Remove**, **Update**, or **Remove File** buttons.

Be aware: When writing this book, we added a new Web module to an already installed enterprise application. When we later selected the Manage Modules link for the application, the new module did not show. Selecting View Deployment Descriptor did not show the new module either. However, when exporting the whole application as an EAR file, the new module was included, so it had in fact been added.

Replacing or adding single files in an application or module

To replace a single file, such as a GIF image or a properties file in an application or module, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application installation window, select the **Replace or add a single file** option.
3. In the **Relative path to file** field, enter the relative path to the file to replace in the EAR file. For example, if you were to replace the logo.gif in the images directory of the HelloWeb.war Web module, you would enter HelloWeb.war/images/logo.gif. If you enter a path or file that does not exist in the EAR file, it will be added.
4. Select either the **Local file system** or **Remote file system** option and click the **Browse** button to locate the updated file. Click **Next**.
5. On the Updating Application window, click **OK**.
6. When the application has been updated in the Master repository, select the **Save** link.
7. If you are working in a distributed server environment, make sure that you also synchronize the changes with the nodes.

Removing application content

Files can also easily be removed either from an EAR file or from a module in an EAR file.

Removing files from an EAR file

To remove a file from an EAR file, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to remove the file from and click the **Remove File** button.
2. In the Remove file dialog box, select the file to be removed and click **OK**.
3. Save the configuration.

Removing files from a module

To remove a file from a module, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications** and click the link for the application to which the module belongs.
2. Click the **Manage Modules** link under the Modules section.
3. Select the module to remove the file from and click the **Remove File** button.
4. In the Remove a file from a module dialog, select the file to be removed and click **OK**.
5. Save the configuration.

Performing multiple updates to an application or module

Multiple updates to an application and its modules can be packaged in a compressed file, .zip, or .gzip format, and uploaded to WebSphere Application Server. The uploaded file is analyzed and the necessary actions to update the application are taken.

Depending on the contents of the compressed file, this method to update an application can replace files in, add new files to, and delete files from the installed application all in one single administrative action. Each entry in the compressed file is treated as a single file, and the path of the file from the root of the compressed file is treated as the relative path of the file in the installed application.

- ▶ To replace a file, a file in the compressed file must have the same relative path as the file to be updated in the installed application.

- ▶ To add a new file to the installed application, a file in the compressed file must have a different relative path than the files in the installed application.
- ▶ To remove a file from the installed application, specify metadata in the compressed file using a file named META-INF/ibm-partialapp-delete.props at any archive scope.

The `ibm-partialapp-delete.props` file must be an ASCII file that lists files to be deleted in that archive with one entry for each line. The entry can contain a string pattern, such as a regular expression that identifies multiple files. The file paths for the files to be deleted must be relative to the archive path that has the `META-INF/ibm-partialapp-delete.props` file.

- ▶ To delete a file from the EAR file (not a module), include a `META-INF/ibm-partialapp-delete.props` file in the root of the compressed file. In the `.props` file, list the files to be deleted. File paths are relative to the root of the EAR file.

For example, to delete a file named `docs/readme.txt` from the root of the `HelloApp.ear` file, include the line `docs/readme.txt` in the `META-INF/ibm-partialapp-delete.props` file in the compressed file.

- ▶ To delete a file from a module in the EAR, include a `module_uri/META-INF/ibm-partialapp-delete.props` file in the compressed file. The `module_uri` part is the name of the module, such as `HelloWeb.war`.

For example, to delete `images/logo.gif` from the `HelloWeb.war` module, include the line `images/logo.gif` in the `HelloWeb.war/META-INF/ibm-partialapp-delete.props` file in the compressed file.

- ▶ Multiple files can be deleted by specifying each file on its own line in the metadata `.props` file.

Regular expressions can also be used to target multiple files. For example, to delete all JavaServer Pages (`.jsp` files) from the `HelloWeb.war` file, include the line `.*jsp` in the `HelloWeb.war/META-INF/ibm-partialapp-delete.props` file. The line uses a regular expression, `.*jsp`, to identify all `.jsp` files in the `HelloWeb.war` module.

As an example, assume we have prepared the compressed `HelloApp_update.zip` file shown in Figure 13.

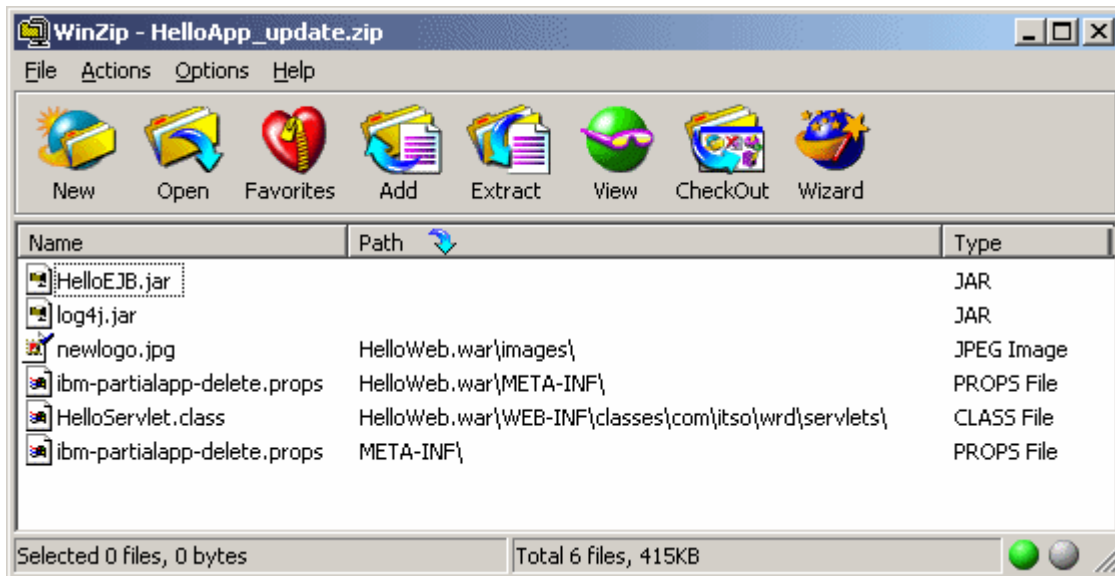


Figure 13 HelloApp_update.zip compressed file

The META-INF/ibm-partialapp-delete.props file contains the following line:

docs/readme.txt

The HelloWeb.war/META-INF/ibm-partialapp-delete.props contains the following lines:

images/logo.gif

When performing the partial application update using the compressed file, WebSphere does the following actions:

- ▶ Adds the log4j.jar file to the root of the EAR.
- ▶ Updates the entire HelloEJB.jar module.
- ▶ Deletes the docs/readme.txt file (if it exists) from the EAR file, but not from any modules.
- ▶ Adds the images/newlogo.jpg file to the HelloWeb.war module.
- ▶ Updates the HelloServlet.class file in the WEB-INF/classes/com/itso/wrdservlets directory of the HelloWeb.war module.
- ▶ Deletes the images/logo.gif file from the HelloWeb.war module.

To perform the actions specified in the HelloWeb_updated.zip file, do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application installation window, select the **Replace, add, or delete multiple files** option.
3. Select either the **Local file system** or **Remote file system** option and click the **Browse** button to select the compressed ZIP file with the modifications you have created. Click **Next**.
4. On the Updating Application window, click **OK**.
5. When the application has been updated in the Master repository, select the **Save To Master Configuration** link.
6. If in a distributed server environment, make sure the **Synchronize changes with Nodes** option is selected so that the application is distributed to all nodes. Click the **Save** button. The application is distributed to the nodes, updated, and restarted as necessary.
7. If the application update changes the set of URLs handled by the application (servlet mappings added, removed or modified), make sure the Web server plug-in is regenerated and propagated to the Web server.

Rolling out application updates to a cluster

The Rollout Update feature allows you to easily roll out a new version of an application, or part of an application using the techniques described previously, to a cluster. The Rollout Update feature takes care of stopping the cluster members, distributing the new application, synchronizing the configuration, and restarting the cluster members. The operation is done sequentially over all cluster members in order to keep the application continuously available.

When stopping and starting the cluster members, the Rollout Update feature works on node level, so all cluster members on a node are stopped, updated, and then restarted, before the process continues to the next node.

Because the Web server plug-in module is not able to detect that an individual application on an application server is unavailable, the Rollout Update feature always restarts the whole application server hosting the application. Because of this, if HTTP session data is critical to your application, it should either be persisted to database or replicated to other cluster members using the memory-to-memory replication feature.

The order in which the nodes are processed and the cluster members are restarted is the order in which they are read from the cell configuration repository. There is no way to tell the Rollout Update feature to process the nodes and cluster members in any particular order.

Assume that we have an environment with two nodes, ITSOBankNode1 and ITSOBankNode2, and a cluster called ITSOBankCluster, which has one cluster member on each node (ITSOBankServer1 on ITSOBankNode1 and ITSOBankServer2 on ITSOBankNode2). Assume we have an application called RAD75EJBWebEAR deployed and running on the cluster.

To update this application using the Rollout Update feature, we would do the following steps:

1. Select **Applications** → **Application Types** → **WebSphere enterprise applications**. Select the application to update and click the **Update** button.
2. On the Preparing for the application installation window, select the appropriate action depending on the type of update. In this example, we will update the entire application EAR to a new version, so we select the **Replace the entire application** option.
3. Select either the **Local file system** or **Remote file system** option and click the **Browse** button to select the updated EAR file. Click **Next**.
4. Proceed through the remaining windows and make any changes necessary. For information about the windows, see “Deploying the application” on page 20. On the Summary window, click **Finish**.

5. When the application has been updated in the master repository, the status window shown in Figure 14 is displayed.

ADMA5013I: Application RAD75EJBWebEAR installed successfully.

Application RAD75EJBWebEAR installed successfully.

If you want to do a rolling update of the application on the cluster(s) on which it is installed, then click Rollout Update. A rolling update will save all changes made in this session to the master configuration, then synchronize and recycle the cluster members on each node, one node at a time.

[Rollout Update](#)

To start the application, first save changes to the master configuration.

The application might not be immediately available while being started on all servers.

Changes have been made to your local configuration. You can:

- [Save](#) directly to the master configuration.
- [Review](#) changes before saving or discarding.

To work with installed applications, click the "Manage Applications" link.

[Manage Applications](#)

Figure 14 Preparing for application rollout

You then have two options to start the rollout action:

- Click the **Rollout Update** link.
- Click the **Manage Applications** link and on the Enterprise Applications window, select the application and click the **Rollout Update** button.

Note: Do not click the **Save directly to the master configuration** link or otherwise save the configuration yourself. The Rollout Update will do that for you. If you save the configuration yourself, the rollout update action will be canceled and it will be handled as a normal application update.

During the rollout, the window in Figure 15 is displayed in the status window.

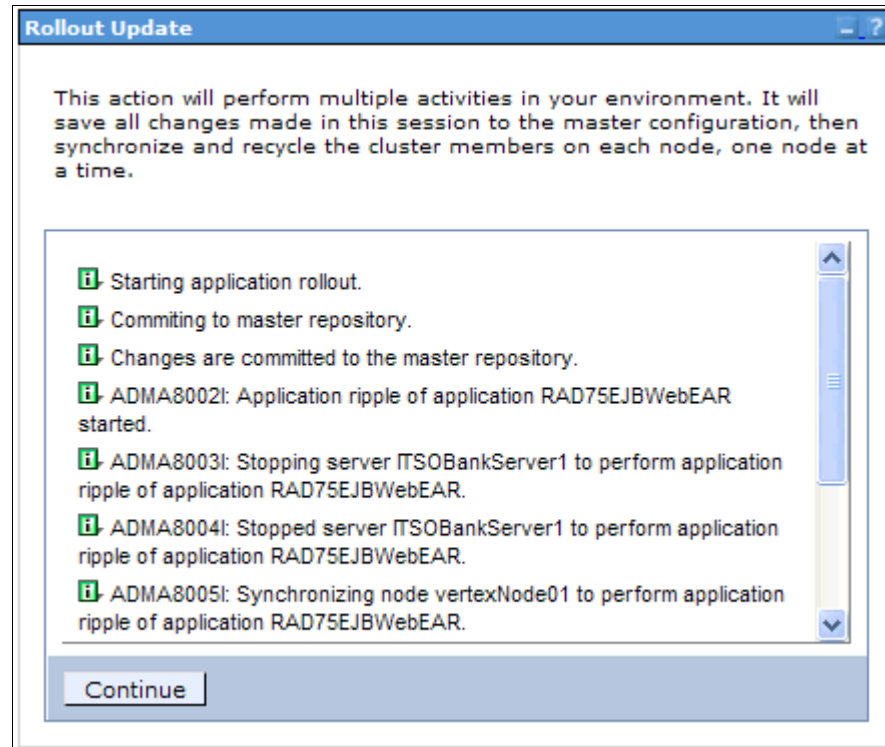


Figure 15 Rolling out an application

For each node, the cluster members are stopped, the application is distributed, and they are restarted. When the rollout has completed (the last message says “The application rollout succeeded”, click **Continue**.

6. If the application update changes the set of URLs handled by the application (servlet mappings added, removed, or modified), make sure the Web server plug-in is regenerated and propagated to the Web server.

Note: The automatic file synchronization of the node agent is temporarily disabled during the rollout process and then re-enabled afterwards, if it was previously enabled. The Rollout Update feature works regardless of the automatic file synchronization setting. However, in production systems, the automatic synchronization is often disabled anyway to give the administrator greater control over exactly when changes made to the cell configuration are distributed to the nodes.

Although the Rollout Update feature makes it very easy to roll out an application to a cluster while keeping the application continuously available, make sure that your application can handle the roll out.

For example, assume you have version 1.0 of an application running in a cluster consisting of two application servers, server1 and server2, and that HTTP session data is persisted to a database. When you roll out version 2.0 of the application and server1 is stopped, the Web server plug-in redirects the users on server1 to server2. Then, when server1 is started again, bringing up version 2.0 of the application, the plug-in will start distributing requests to server1 again. Now, if the application update incurred a change in the interface of any class stored in the HTTP session, when server1 tries to get these session objects from the database, it might run into a deserialization or class cast exception, preventing the application from working properly.

Another situation to consider is when the database structure changes between application versions, as when tables or column names change name or content. In that case, the whole application might need to be stopped and the database migrated before the new version can be deployed. The Rollout Update feature would not be suitable in that kind of scenario.

So it is very important to understand the changes made to your application before rolling it out.

WebSphere Virtual Enterprise: If you want even more advanced rollout and application versioning capabilities than what is available in WebSphere Application Server, take a look at the WebSphere Virtual Enterprise product at:

<http://www.ibm.com/software/webservers/appserv/extend/virtualenterprise/>

WebSphere Virtual Enterprise extends an existing WebSphere infrastructure and brings, among many other features, functions to validate new versions of applications in production environments, and then to roll them out seamlessly, with features to drain application servers from existing users before taking them offline for the update.

Hot deployment and dynamic reloading

Hot deployment and dynamic reloading characterize how application updates are handled when updates to the applications are made by directly manipulating the files on the server. In either case, updates do not require a server restart, though they might require an application restart:

▶ Hot deployment of new components:

Hot deployment of new components is the process of adding new components, such as WAR files, EJB JAR files, EJBs, servlets, and JSP files to a running application server without having to stop and then restart the application server.

However, in most cases, such changes require the application itself to be restarted, so that the application server runtime reloads the application and its changes.

▶ Dynamic reloading of existing components:

Dynamic reloading of existing components is the ability to change an existing component without the need to restart the application server for the change to take effect. Dynamic reloading can involve changes to the:

- Implementation of an application component, such as changing the implementation of a servlet
- Settings of the application, such as changing the deployment descriptor for a Web module

To edit the files manually, locate the binaries in use by the server (see Chapter 14, *Packaging applications for deployment*). Although the application files can be manually edited on one or more of the nodes, these changes will be overwritten the next time the node synchronizes its configuration with the deployment manager. Therefore, we recommend that manual editing of an application's files should only be performed in the master repository, located on the deployment manager machine.

Note: Unless you are familiar with updating applications by directly manipulating the server files, it might be better to use the administrative console Update wizard.

There are three settings that affect dynamic reload:

- ▶ Reload classes when application files are updated:

In order for application files to be reloaded automatically after an update, the **Override class reloading settings for Web and EJB modules** setting must be enabled and the **Polling interval for updated files** setting must be greater than 0.

Select **Applications** → **Application Types** → **WebSphere enterprise applications**, and click the link for the application. In the Detail properties section, click the **Class loading and update detection** link.

- ▶ Application Server class loader policy:

The application server's class loader policy should be set to Multiple. If it is set to Single, the application server will need to be restarted after an application update.

Select **Servers** → **Server Types** → **WebSphere application servers**, and click the server name. The setting is found in the General Properties section.

- ▶ JSP Reload options for Web modules:

A Web container reloads a Web module only when this setting is enabled.

Select **Applications** → **Application Types** → **WebSphere enterprise applications**, and click the link for the application. In the Web Module Properties section, click the **JSP and JSF options**, and then select the **JSP enable class reloading** option and enter a polling interval.

For more information about using hot deployment and dynamic reload, see the topics, *Updating applications* and *Hot deployment and dynamic reloading*, in the Information Center.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

© Copyright International Business Machines Corporation 2009. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This document REDP-4583-00 was created or updated on October 13, 2009.



Send us your comments in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:
ibm.com/redbooks
- ▶ Send your comments in an email to:
redbook@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099, 2455 South Road
Poughkeepsie, NY 12601-5400 U.S.A.




Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DB2®
IBM®

Rational®
Redbooks®

Redbooks (logo) ®
WebSphere®

The following terms are trademarks of other companies:

Interchange, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

EJB, J2EE, Java, JavaServer, JDBC, JDK, JNI, JRE, JSP, JVM, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

ActiveX, Visual Basic, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.