WebSphere® WebSphere Everyplace Deployment for Windows and Linux

Version 6.0

System Administrator's Guide

Note

Before using this information and the product it supports, read the information in the Notices section.

Note

The PDF version of this guide is provided for your convenience. The help system is the preferred way to read the documentation.

First Edition (October, 2005)

This edition applies to version 6.0 release and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Welcome to the WebSphere Everyplace Deployment for Windows and Linux System Administrator's Guide

The following table lists some of the tasks you might perform as a system administrator for the WebSphere Everyplace Deployment clients, and the topics that help you complete these tasks.

Tasks	**Topics**
Install the client	"Installation" on page 9
Configure the client	"Managing client runtimes" on page 23
Manage the client	"Configuring security" on page 71
Change the platform branding	"Specifying platform branding" on page 32
Uninstall the client	"Uninstalling WebSphere Everyplace Deployment" on page 22
Install, update, manage, and uninstall applications	• "Managing applications with WebSphere Everyplace Deployment" on page 55 • "Managing applications with a Device Management Server" on page 63

Introduction

WebSphere Everyplace Deployment for Windows® and Linux® enables you to develop powerful applications that run on desktop and laptop clients across Windows and Linux operating systems. Using the WebSphere Everyplace Client Toolkit, application developers can leverage their experience with developing J2EE applications, Web Services, or Eclipse applications, to develop applications that run on WebSphere Everyplace Deployment for Windows and Linux.

With WebSphere Everyplace Deployment for Windows and Linux (also called *the client platform*), you can move key components of your applications from the server to desktop and laptop clients by using standard APIs and services.

Moving application components to run on a client can have dramatic results for business. End-users benefit from improved application response time because applications perform business operations locally on the client. As a result, there is a reduction in network traffic between clients and servers, and in server workload. Furthermore, mobile end-users can continue to productively use their applications from their clients even when they are at a location that does not have network connectivity, such as a customer site. You can also utilize the local graphical user interface (GUI) capabilities of the client devices to deliver a richer user experience than can be supported by a Web browser.

The client platform provides a way to install, launch and manage multiple applications within an integrated application desktop window. The client platform is built on the Eclipse 3.0.2 Rich Client Platform (RCP) framework. For application developers familiar with the Eclipse Integrated Development Environment, or Rational® tools, it is the same core platform with the application development tools removed, leaving a basic container for features and components, and a window management and layout environment.

The client platform provides the following set of standards-based Client Services for the development of your client applications:

- **Managed Client Services** including a JVM, a robust component framework, and additional component services.
- **Platform Management** including the Eclipse Update Manager and an Enterprise management agent to install and update applications and services on the client platform.
- **Access Services** including relational database services and synchronization, transactional messaging, Web Services, a Web container to run local Web applications, an embedded transaction container to run local embedded Enterprise Java™ Beans (EJB's), and more.
- **Interaction Services** including integrated browser controls to launch Web applications, the Standard Widget Toolkit (SWT) and JFace Toolkit to support GUI applications, and a Workbench that enables end-users to install and launch one or more applications.

The client platform provides a standard set of APIs, such as JDBC, and JMS, that you use to invoke these services.

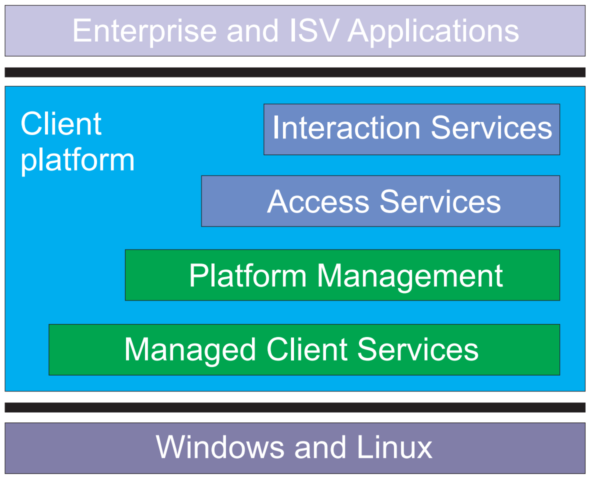

Figure 1. The client platform

The WebSphere® Everyplace® Client Toolkit (also called *the toolkit*) provides a complete, integrated set of tools that allows you to develop, debug, test, package and deploy client applications that use the Client Services. This toolkit is built on Eclipse technology and extends the powerful Rational suite of development tools so you can leverage your existing skills and software components. The toolkit also provides Ant tasks so you can create Ant scripts to automate the building of your applications. In addition, the toolkit provides program samples to help you quickly start your application development projects.

The combination of the WebSphere Everyplace Deployment for Windows and Linux client and the WebSphere Everyplace Deployment server (also called *the server platform*) provide the client and server middleware necessary to deliver and manage end-to-end applications (see figure 1). Administrators use the WebSphere Everyplace Deployment server to install and configure the server middleware, so client applications can securely perform assured transactions and database synchronization with enterprise applications and data. For more information on the server platform, please refer to the WebSphere Everyplace Deployment server documentation.

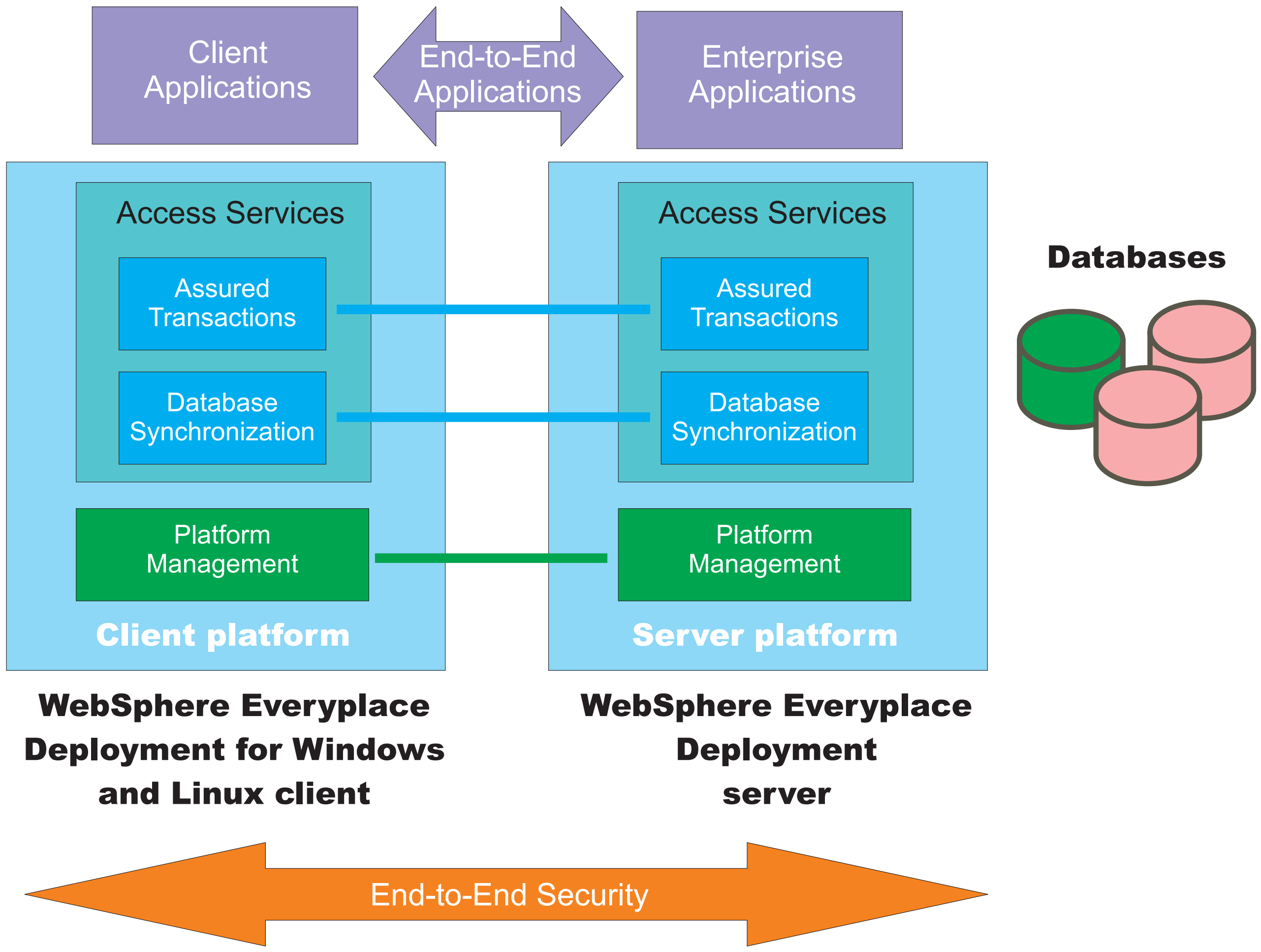

Figure 2. End-to-end solution

Packaging

Client applications and application services are packaged as features, each of which consists of one or more components. The client platform cannot directly run J2EE packaging artifacts such as EAR and WAR files.

Components

The Eclipse framework, and therefore the client platform, is organized around a plug-in and extension point model. The framework provides a core set of components. Additional components are provided in a directory or JAR file organized in a specific structure, and implement instantiations of the various extension points. The framework reads the component declarative information, and incorporates the components into the correct locations in the framework.

A plug-in is the level at which components are declared to the Eclipse framework. A plug-in is a JAR file with a plug-in manifest file named `plugin.xml`. The plug-in manifest describes the plug-in to the framework and enables a plug-in to consume and/or provide extensions from/to other plug-ins. For example, a plug-in can provide user interface extensions, such as perspectives, views, editors, and wizards. It can also provide business logic or core services to other plug-ins, but contribute no extensions to the user interface.

A bundle is the level at which components are declared to the OSGi Service Framework. A bundle is a JAR file with a bundle manifest file named `MANIFEST.MF`. The bundle manifest describes the bundle to the service framework and enables a bundle to consume and/or provide packages and services from/to other bundles. Bundles can also include a Bundle Activator class. The Bundle-Activator header in the bundle manifest file identifies the class to the framework. At startup time, the framework creates an instance of this class and calls its `start()` method. The Bundle Activator can then publish services, start its own threads, and so on. When the bundle shuts down, the framework calls the activator's `stop()` method. While the bundle shuts down, the Bundle Activator can release resources that are obtained since the start method was called and revoke any services it has published.

Recall that the Eclipse framework is built on the OSGi Service Framework. Therefore, you can define each component in your applications as a plug-in, a bundle, or both depending on your requirements.

Note: For a component to be recognized by the client platform, the toolkit and the Eclipse Plug-in Development Environment (PDE), it must have a unique name and version. If you develop a plug-in, specify a unique value for the name attribute and a version number for the version attribute in the plug-in manifest. If you develop a bundle, specify a unique value for the `Bundle-SymbolicName` attribute and a version number for the `Bundle-Version` attribute in the bundle manifest.

A component can generally be organized in one of three ways:

- A directory containing at least a `plugin.xml` file. The directory may also contain a `MANIFEST.MF` file located in the `META-INF` directory, additional files, as well as Java code contained within JAR files.

  A `plugin.xml` file is required if the component defines extension points for use by other plug-ins or implements extension points provided by other plug-ins.
- A directory containing at least a `MANIFEST.MF` file in the `META-INF` directory. The directory will also contain Java code contained in JAR files. The `MANIFEST.MF` will refer to the JARs by referencing them via the `Bundle-Classpath` attribute.

  Components that provide only business logic or OSGi services and do not intend to provide or implement any extension points can use this format. Components without `plugin.xml` files that need to be available when building other components or when launching the client platform by using either the toolkit or the PDE must be organized in this format.
- A single JAR file containing at least a `META-INF\MANIFEST.MF` or a `plugin.xml` file.

  Components may be provided for use in the client platform by collecting all of the component artifacts into a single JAR file. While this organization will run successfully, this organization is not compatible with the toolkit and Eclipse PDE.

Fragments

A component may not always provide a complete implementation. In some cases, fragments may be used to complete or extend a component.

For example, the primary component may provide an implementation that contains translatable text in a default language. Fragments are then used to provide translations for additional languages.

A second case where fragments are often used is to provide platform (processor/operating system) specific implementations.

Fragments contain either a `fragment.xml` (similar to a `plugin.xml`), or a `MANIFEST.MF`, or both. A fragment is associated with, or dependent upon, a specific primary component, but still maintains a unique identity. Querying a list of components will also return fragments, so that these fragments can be individually started and stopped.

Fragments generally add classes or resources to the class path normally used by the primary component. Fragments do not contain Bundle-Activator classes. Since fragments are only extensions to a component, they cannot be required or imported by another component.

Features

On disk, an Eclipse-based product is structured as a collection of components and fragments. Each component or fragment contains the code that provides some of the product's functionality. The code and other files for a component or fragment are installed on the local computer, and get activated automatically as required. A product's components are grouped together into features. A feature is the smallest unit of separately downloadable and installable functionality

The fundamentally modular nature of the Eclipse platform makes it easy to install additional features and components into an Eclipse-based product, and to update the product's existing features and components. You can do this either by using traditional native installers running separately from Eclipse, or by using the Eclipse platform's own Update Manager. The Eclipse Update Manager can be used to discover, download, and install updated features and components from special web based Eclipse update sites.

The basic underlying mechanism of the Update Manager is simple: the files for a feature or component are always stored in a sub-directory whose name includes a version identifier (e.g., "2.0.0"). Different versions of a feature or component are always given different version identifiers, thereby ensuring that multiple versions of the same feature or component can co-exist on disk. This means that installing or updating features and components requires adding more files, but never requires deleting or overwriting existing files. Once the files are installed on the local computer, the new feature and component versions are available to be configured. The same installed base of files is therefore capable of supporting many different configurations simultaneously; installing and upgrading an existing product is reduced to formulating a configuration that is incrementally newer than the current one. Important configurations can be saved and restored to active service in the event of an unsuccessful upgrade.

Large Eclipse-based products can organize their features into trees starting from the root feature that represents the entire product. This root feature then includes smaller units of functionality all the way down to leaf features that list one or more plug-ins and fragments. The capability to group features hierarchically allows products to be built on top of smaller products by including the smaller products and adding more features.

Some included features may be useful add-ons but not vital to the proper functioning of the overall product. Feature providers can elect to mark these features as optional. The Update Manager allows users to choose whether or not to install optional features. If not installed right away, optional features can be added at a later date.

Platform management

Platform Management installs, maintains, and configures applications and services on the client. There are two platform management services.

The Update Manager enables end-users to directly install applications and components from standard Eclipse update sites onto the Workbench.

The Enterprise Management Agent works cooperatively with the Device Management Server provided by the WebSphere Everyplace Deployment server to perform management operations. The agent and server use the SyncML/DM protocol defined by the Open Mobile Alliance to communicate management requests. An administrator can schedule management jobs for devices that include software installation, update, and configuration. When installing and updating software components, the management system determines which components are already on the device and then installs only the missing components.

Installation

This section describes how to install WebSphere Everyplace Deployment, version 6.0 onto client machines. There are several installation methods available. Before you install WebSphere Everyplace Deployment ensure that each machine on which you are installing meets the prerequisites.

Client prerequisites

Ensure that each machine on which you plan to install WebSphere Everyplace Deployment meets these requirements.

Minimum hardware requirements include the following for the client machines:

- 256MB RAM
- 150 MB free disk space
- x86 processors capable of supporting Windows XP

Windows IA32 and Linux IA32 environments are supported as follows:

- Microsoft® Windows XP Service Pack 1
- Microsoft Windows XP Service Pack 2
- RedHat ® EL 3.0 WS with GTK support – Update 3

Note: Microsoft Windows 2000 Professional (any service pack) is not supported.

Related information

"Installing from a CD" on page 10
Before you install WebSphere Everyplace Deployment, check that there is enough space in your operating system's temporary directory. In addition to the space required to install the product, your operating system's temporary directory must have at least 100 MB free.

"Invoking a silent installation" on page 11
You can set up the WebSphere Everyplace Deployment installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file, you can customize the silent installation to automatically select panel options and installation properties.

"Migrating from WCTME 5.8.1" on page 15
IBM® WebSphere Everyplace Deployment 6.0 can coexist with IBM Workplace™ Client Technologies, Micro Edition - Enterprise Offering (WCTME EO) 5.8.1. For coexistence, you can install into a different directory. User settings and applications can be migrated from WCTME EO 5.8.1 to WebSphere Everyplace Deployment 6.0.

"Setting up a web download site" on page 17
You can set up WebSphere Everyplace Deployment on a Web environment, such as an HTTP server, so that users can perform one-click download and installation.

Installing from a CD

Before you install WebSphere Everyplace Deployment, check that there is enough space in your operating system's temporary directory. In addition to the space required to install the product, your operating system's temporary directory must have at least 100 MB free.

Use the following steps to install WebSphere Everyplace Deployment from a CD.

1. Insert the WebSphere Everyplace Deployment CD into the CD-ROM drive.
2. If autorun is enabled on your system, a browser opens and displays a welcome page. If autorun is disabled on your system, launch the `index.html` file from the root of the installation CD.
3. Launch the appropriate installer executable for your system. Installer executables are located in the `install/`directory of the CD.
   a. On a Windows system, run the `setupwin32.exe` file.
   b. On a Linux system, run the `setuplinux.bin` file.

   Note:
   - If a File Download dialog displays, click **Open** to run the installer.
   - You can run the installer directly from the `install/` directory on the installation CD instead of the browser.
   - If you want to install WebSphere Everyplace Deployment from a console or telnet terminal without using the graphical interface, run the following command from the first installation CD:

     ```
     installer executable -console
     ```

     where *installer executable* is the name of the installer executable file.
   - If you want to use a screen reader to assist you with installation, specify the `-accessibility` option on the command line as follows:

     ```
     installer executable -accessibility
     ```

     where *installer executable* is the name of the installer executable file.
4. The WebSphere Everyplace Deployment Installer guides you through the rest of the installation:
   a. At the installer welcome page, click **Next**.
   b. Read the license. Select **I accept the terms in the license agreement** if you are willing to accept the license and click **Next**.
   c. Enter the destination where you want to install the WebSphere Everyplace Deployment client and click **Next**.

      Notes:
      - The default destination may not be accessible by the current user. In this case, change the location to a user-accessible destination, for example `/<user_home> /IBM/WED`.
      - You can not install the product into a location where another instance of this product already exists. The installer will instruct you to go back and provide a different destination.

   d. Select the setup type and click **Next**.
   e. If you want to configure the Enterprise Management Agent, select **Yes**, otherwise, select **No**. Click **Next** to continue. For more information on configuring the Enterprise Management Agent, refer to "Configuring the Enterprise Management Agent" on page 41.

f. If you selected to configure the Enterprise Management Agent, proceed to the next step, otherwise proceed to step 4h.
g. Provide settings for the Enterprise Management Agent and click **Next**.
h. Before the actual installation begins, verify the installation summary information is correct. Click **Next** to begin the installation.
i. Click **Next** to provision additional platform components.
j. If WCTME EO 5.8.1 was detected on your system, proceed to the next step, otherwise, proceed to step 4m.
k. If you want to migrate data from WCTME EO 5.8.1 to WED 6.0. Select **Yes** to migrate, otherwise, select **No**. Click **Next** to continue. Refer to "Migrating from WCTME 5.8.1" on page 15 for more information.
l. If you wish to uninstall WCTME EO 5.8.1. Select **Yes** to uninstall, otherwise select **No**. Click **Next** to continue.
m. If you want to launch the platform after installation is complete, select **Yes**, otherwise select **No**. Click **Next** to continue.
n. If you selected to launch the platform, click **Next** to close the installer and launch the platform, otherwise, click **Finish**.

5. After the installation completes, shortcuts are created:
   - On Windows systems, a shortcut to launch the platform displays on the desktop and an IBM WebSphere Everyplace Deployment program folder containing shortcuts is created in **Start > Programs**.
   - On Linux systems, shortcuts will be placed in the same start menu **Office** folder.

Related information

"Client prerequisites" on page 9
Ensure that each machine on which you plan to install WebSphere Everyplace Deployment meets these requirements.

"Invoking a silent installation"
You can set up the WebSphere Everyplace Deployment installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file, you can customize the silent installation to automatically select panel options and installation properties.

"Setting up a web download site" on page 17
You can set up WebSphere Everyplace Deployment on a Web environment, such as an HTTP server, so that users can perform one-click download and installation.

Invoking a silent installation

You can set up the WebSphere Everyplace Deployment installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file, you can customize the silent installation to automatically select panel options and installation properties.

To install silently with default options:

1. From a command line, change to the `install/` directory on the CD.
2. Run the following command:
   ```
   installer executable -silent -G licenseAccepted=true
   ```
   Where *installer executable* is the installation executable file name.

To install silently with custom options:

1. Create an options file using the steps provided in the "Using options files" on page 16 section.
2. Modify the options file with the customized values for your installation.
3. From a command line, change to the `install/` directory on the installation CD.
4. Run the following command:

   ```
   installer executable - silent -options option file
   ```

   Note: When using an options file to install silently, you must set the following property in your options file for the installation to proceed:

   ```
   -G licenseAccepted=true
   ```

Related tasks

"Installing for system management"
You can install the WebSphere Everyplace Deployment platform to run as a system management service. This configuration enables a system administrator to manage the entire system with the platform's management agent that communicates with the Device Management Server. In this configuration, there is no user interface to interact with the platform. Configuration of the management agent settings is accomplished either during the installation process or afterwards by using an agent configuration servlet that you access from a browser.

Related information

"Client prerequisites" on page 9
Ensure that each machine on which you plan to install WebSphere Everyplace Deployment meets these requirements.

"Installing from a CD" on page 10
Before you install WebSphere Everyplace Deployment, check that there is enough space in your operating system's temporary directory. In addition to the space required to install the product, your operating system's temporary directory must have at least 100 MB free.

"Migrating from WCTME 5.8.1" on page 15
IBM WebSphere Everyplace Deployment 6.0 can coexist with IBM Workplace Client Technologies, Micro Edition - Enterprise Offering (WCTME EO) 5.8.1. For coexistence, you can install into a different directory. User settings and applications can be migrated from WCTME EO 5.8.1 to WebSphere Everyplace Deployment 6.0.

"Setting up a web download site" on page 17
You can set up WebSphere Everyplace Deployment on a Web environment, such as an HTTP server, so that users can perform one-click download and installation.

Installing for system management

You can install the WebSphere Everyplace Deployment platform to run as a system management service. This configuration enables a system administrator to manage the entire system with the platform's management agent that communicates with the Device Management Server. In this configuration, there is no user interface to interact with the platform. Configuration of the management agent settings is accomplished either during the installation process or afterwards by using an agent configuration servlet that you access from a browser.

To install WebSphere Everyplace Deployment client for system management, you must have system administrator or root privileges on the system.

1. Insert the WebSphere Everyplace Deployment CD into the CD-ROM drive.

2. If autorun is enabled on your system, a browser opens and displays a welcome page. If autorun is disabled on your system, launch the `index.html` file from the root of the installation CD.
3. Launch the appropriate installer executable for your system. Installer executables must be run from the `install/` directory of the CD.
   a. On a Windows system, run the `setupwin32_service.bat` file.
   b. On a Linux system, run the `setuplinux_service.sh` file.

   Note:
   - If you want to install WebSphere Everyplace Deployment from a console or telnet terminal without using the graphical interface, run the following command from the first installation CD:

     ```
     installer executable -console
     ```

     where *installer executable* is the name of the installer executable file.
   - If you want to use a screen reader to assist you with installation, specify the `-accessibility` option on the command line as follows:

     ```
     installer executable -accessibility
     ```

     where *installer executable* is the name of the installer executable file.
4. The WebSphere Everyplace Deployment Installer guides you through the rest of the installation:
   a. At the installer welcome page, click **Next**.
   b. Read the license. Select **I accept the terms in the license agreement** if you are willing to accept the license and click **Next**.
   c. Enter the destination where you want to install the WebSphere Everyplace Deployment client and click **Next**.

      Notes:
      - You can not install the product into a location where another instance of this product already exists. The installer will instruct you to go back and provide a different destination
      - You can not install more than one service on the system at a time. If the installer detects that a service already exists, then the installation will not proceed.
      - On Linux, the destination path must not contain spaces. If the directory path contains spaces, the startup scripts will not run correctly.
   d. Provide a web container port value and click **Next**.
   e. Provide settings for the Enterprise Management Agent and click **Next**.
   f. Before the actual installation begins, verify the installation summary information is correct. Click **Next** to begin the installation.
   g. Click **Next** to provision additional platform components.
   h. If you want to launch the platform after installation is complete, select **Yes**, otherwise select **No**.
   i. If you selected to launch the platform, click **Next** to close the installer and launch the platform, otherwise, click **Finish**.
5. After the installation completes, shortcuts are created.
   - On Windows systems, an IBM WebSphere Everyplace Deployment program folder containing shortcuts is created in **Start > Programs**.
   - On Linux systems, shortcuts will be placed in the same start menu Office folder.

6. If you plan to install native applications using the NativeAppBundle Tool, perform the following additional steps:
   a. **Select Start > Control Panel > Administrative Tools > Services**.
   b. Locate the WED Management Service, and double click it.
   c. Check the **Allow Service to Interact with Desktop** option. and click OK.

   Note: Enabling an interactive desktop may introduce security holes. This makes it possible for another application running on the desktop to interact with the service. Also note that ″Allow Service to Interact with Desktop″ will not work with fast-user switching, Remote Desktop, or Terminal Services. This option can only be utilized by an administrator logging in locally to the target machine.″

If you are installing as a service as part of a silent installation, you must specify the following options, either in the options file (refer to "Using options files" on page 16 for more information), or from the command line:

```
-w webContainerPortPanel.port="<value>"
-w agentSettingsPanel.userName="<value>"
-w agentSettingsPanel.userPassword="<value>"
-w agentSettingsPanel.verifyUserPassword="<value>"
-w agentSettingsPanel.serverURL="<value>"
```

Note: Because the service is running as ″Local System″ on Windows and as ″root″ on Linux, you must make sure that the entire installation tree has the appropriate access rights.

Related information

"Client prerequisites" on page 9
Ensure that each machine on which you plan to install WebSphere Everyplace Deployment meets these requirements.

"Installing from a CD" on page 10
Before you install WebSphere Everyplace Deployment, check that there is enough space in your operating system's temporary directory. In addition to the space required to install the product, your operating system's temporary directory must have at least 100 MB free.

"Invoking a silent installation" on page 11
You can set up the WebSphere Everyplace Deployment installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file, you can customize the silent installation to automatically select panel options and installation properties.

"Migrating from WCTME 5.8.1" on page 15
IBM WebSphere Everyplace Deployment 6.0 can coexist with IBM Workplace Client Technologies, Micro Edition - Enterprise Offering (WCTME EO) 5.8.1. For coexistence, you can install into a different directory. User settings and applications can be migrated from WCTME EO 5.8.1 to WebSphere Everyplace Deployment 6.0.

"Setting up a web download site" on page 17
You can set up WebSphere Everyplace Deployment on a Web environment, such as an HTTP server, so that users can perform one-click download and installation.

Migrating from WCTME 5.8.1

IBM WebSphere Everyplace Deployment 6.0 can coexist with IBM Workplace Client Technologies, Micro Edition - Enterprise Offering (WCTME EO) 5.8.1. For coexistence, you can install into a different directory. User settings and applications can be migrated from WCTME EO 5.8.1 to WebSphere Everyplace Deployment 6.0.

WebSphere Everyplace Deployment 6.0 provides two methods for migrating data and applications from Workplace Client Technology™, Micro Edition - Enterprise Offering (WCTME EO) 5.8.1 — during the WebSphere Everyplace Deployment installation process, or after the installation process from a command line utility.

Note: Migration of Enterprise Management Agent settings or application configurations that make use of the Configuration Admin service is not supported.

Migrating during installation

If the WebSphere Everyplace Deployment installation program finds an installation of WCTME EO 5.8.1 on the system, then the installation program presents migration panels during the WebSphere Everyplace Deployment installation process. The panels show where the WCTME EO 5.8.1 installation was found and asks you if you want to migrate data to WebSphere Everyplace Deployment 6.0. If you choose to migrate, then when the migration process is complete, the installation prompts you if you would like to uninstall the WCTME EO 5.8.1 installation. If you choose to uninstall it, then WCTME EO 5.8.1 will be silently uninstalled. All WCTME EO 5.8.1 content will be removed, including user data.

Note: The WebSphere Everyplace Deployment installation program will not allow you to migrate WCTME EO 5.8.1 data if you installed WebSphere Everyplace Deployment as a system management service.

If you are migrating during a silent installation, you must specify the following options, either in the options file, or from the command line:

```
-w migrate581Panel.migrate581="<value>"
```

If you want to uninstall after migration during silent installation, you must specify the following option:

```
-w uninstall581Panel.uninstall581="<value>"
```

Migrating after installation

A migration utility named `migration.jar` is included on the CD-ROM in the `install/` directory, and is also installed on the system during installation. When installed during installation, the utility resides in the `<`*`installation directory`*`>/migration/`directory.

The migration utility is a Java program that requires a JVM to be present on the system. The utility is run with the following parameters:

```
java -cp migration.jar com.ibm.pvc.install.migration.Migration
-from <WCTME EO 5.8.1 location> -to <WED 6.0 location [options]>
```

The migration utility works with three locations — configuration, workspace, and applications. The default targets for these locations are:

- applications `<user.home>/IBM/RCP/<rcp.install.id>/<user.name>/migrated_apps`
- configuration `<user.home>/IBM/RCP/<rcp.install.id>/.config`
- workspace `<user.home>/IBM/RCP/<rcp.install.id>`

The [options] are optional arguments used to override the default locations used by the migration utility:

```
-apps       <destination applications>
-workspace  <destination workspace>
-config     <destination configuration>
```

Using options files

You use options files to pass command line options to an installation or uninstallation as you would normally specify on the command line to represent the response to the installation wizard panels, and to set installation properties.

The options you specify in the options file are executed after the execution of any options that you enter directly on the command line. You can generate an options file templates from the install or uninstall launcher by running the following command:

```
installer or uninstaller executable -options-template options file
```

Running the command creates an options file with the specified name. You can then modify this file to fill in settings that will be applied when the installation is run.

Note: To create options files for the uninstaller, you must first install the product onto a system. The uninstaller executable is located in `<`*`installation directory`*`>/_uninst/` directory. On Windows systems, the file is named `uninstaller.exe`. On Linux systems, the file is named `uninstaller.bin`.

Recording options

Another approach, instead of manually modifying the template file, is to record an options file based on entering choices during an actual installation. To record an options file, run the following command:

```
installer or uninstaller executable -options-record options file
```

After the installation is complete, an options file with the specified name is automatically generated.

Using the options file during installation

After modifying the template, or recording an options file on installation, run the following command to pass the options to an installation:

```
installer or uninstaller executable -options options file
```

Related information

"Client prerequisites" on page 9
Ensure that each machine on which you plan to install WebSphere Everyplace Deployment meets these requirements.

"Invoking a silent installation" on page 11
You can set up the WebSphere Everyplace Deployment installation program to run silently if you do not want to interact with the installation wizard.

Additionally, using an options file, you can customize the silent installation to automatically select panel options and installation properties.

Setting up a web download site

You can set up WebSphere Everyplace Deployment on a Web environment, such as an HTTP server, so that users can perform one-click download and installation.

Included with the WebSphere Everyplace Client Toolkit is a sample application, called appletInstaller. This sample application provides the ability to deploy the WebSphere Everyplace Deployment runtime from an HTTP server, such as the IBM HTTP Server component of the WebSphere Everyplace Deployment server, to client desktop machines using their Web Browser, and leveraging the browser's JavaScript™ and Java browser capabilities. When the client runtime components are deployed to the machine the appletInstaller launches the Install Shield installer and enables the user to customize the product installation as needed.

The appletInstaller sample application consists of the following files:

- **index.html** - A simple html front-end to allow the user to receive any special instructional information before beginning the download/installation process
- **download.html** - The entry point that analyzes the client's browser and operating system settings and generate the applet tag passing in the information collected to the applet.
- **downloadApplet.js** - Contains utility JavaScript functions
- **filesList.props** - Contains a list of files to be downloaded
- **token-values.props** - Contains definitions for tokens that are used in filesList.props for host and host-name variables.
- **resources directory**- An optional directory that contains: Bitmaps (for branding the applet), resource files (for branding the applet strings), and the signed applet archives Installer.jar and Installer.cab.

The JavaScript in the `download.html` file collects the client's browser and operating system information. If Java is enabled on the client's browser, then the `buildAppletTag()` method is called to download the appletInstaller. The applet contacts the HTTP server to get the `filesList.props`, file which lists the files to be downloaded to the client. The applet then starts the download of the appropriate installer for the client's platform. After downloading the installer, the applet launches the installer and exits.

If Java is not enabled in the client's browser, then the `buildInstallDownload()` method is called. This builds the path to the appropriate installer for the client's platform, and lets the browser handle the download. The collection of files needed for installation can be zipped into one archive file and delivered via standard browser file download capabilities and then manually unzipped and launched by the user to begin the client runtime installation

Supported client platforms

Operating Systems:

- Windows (Microsoft Windows XP SP1 and SP2)
- Linux (RHEL 3 update 3)

Browsers:

- Internet Explorer (Windows)

- Mozilla (Windows, Linux)
- Firefox (Windows, Linux)

Java Virtual Machines:
- Microsoft JVM (Windows IE)
- IBM JVM (Windows IE, Windows Mozilla/Firefox, Linux Mozilla/Firefox)
- Sun JVM (Windows IE, Windows Mozilla/Firefox, Linux Mozilla/Firefox)

JavaScript has been tested with Internet Explorer, Mozilla, and Firefox on both Windows and Linux.

Related information

"Client prerequisites" on page 9
Ensure that each machine on which you plan to install WebSphere Everyplace Deployment meets these requirements.

Installing and running the appletInstaller sample

1. Create a new directory, for example `appletInstaller`, off of the `DocumentRoot` of your HTTP server. For the WebSphere Everyplace Deployment server, the server root defaults to `C:\Program Files\IBM Http Server\htdocs\<lang>` where `<lang>` is the locale string such as en_US for United States English. Thus, for a US English installation of the WebSphere Everyplace Deployment server, the files would be placed in a new directory called `C:\Program Files\IBM Http Server\htdocs\en_US\appletInstaller`. In order to simplify references to this location, this directory will be referred to this directory as `$DocumentRoot/appletInstaller$` .
2. Copy the following files and directories from your installation media to the `$DocumentRoot/appletInstaller$` directory.
   a. All files from the `<installation_media>/install/applet/bin/` directory
   b. All files from the `<installation media>/updates` directory
   c. The following files from the `<installation media>/install` directory: **setupwin32.exe**, **setuplinux.bin**, **setup.jar**
   d. All files from the `<installation media>/install/deploy/` directory and its contents

   After completing the copy of these files and directories, the `$DocumentRoot/appletInstaller$` directory should look similar to the following:
   ```
   index.html
   download.html
   downloadApplet.js
   filesList.props
   token-values.props
   WebInstaller.jar
   WebInstaller.cab
   setupwin32.exe
   setup.jar
   setuplinux.bin
   updates/    (directory with the same contents as on the installation media)
   deploy/     (directory with the same contents as on the installation media)
   ```
3. Edit `$DocumentRoot/appletInstaller$/deploy/install_X.properties`, where X is the number 0,1, or 2, and replace all occurrences of `INSTALLER_HOME/../updates/` with the URL of the `updates/` directory on your HTTP server, such as `http://<IP_ADDR>/appletInstaller/updates`.
4. If you want to support non-Java enabled browsers you will need to complete the following steps:

a. Create a new file called `setupwin32.zip` in the `$DocumentRoot/appletInstaller$` directory by zipping up the following files from the new HTTP server directory: `setupwin32.exe`, `setup.jar`, and the deploy directory and its contents.
b. Create a new file called `setuplinux.zip` in the `$DocumentRoot/appletInstaller$` directory by zipping up the following files from the new HTTP server directory: `setuplinux.bin`, `setup.jar`, and the deploy directory and its contents.

5. Optionally: The configuration files default to `$host$/appletInstaller` as the location of the appletInstaller on your HTTP server. Thus, if you used the example directory, you do not need to make any changes in the `$DocumentRoot/appletInstaller$/filesList.props` file. If you would like to change the default installation location you can edit the `$DocumentRoot/appletInstaller$/filesList.props` file and update all of its `.location` entries to reflect the external path on your HTTP server to the directory that you have copied the files in steps 1 and 2. For example, if you placed the files in a directory called `appletInstaller2` directly off of your HTTP server root, then all of the `.location` entries should be set to `$host$/appletInstaller2` .
6. Optionally: If you want the appletInstaller to silently install the WebSphere Everyplace Deployment for Windows and Linux client with a predefined set of configuration parameters, follow the steps in "Invoking a silent installation" on page 11 and "Using options files" on page 16. These sections will help you create the necessary options file, which we refer to as `optionsFile.txt`. Once you have created an options file, place the file into the `$DocumentRoot/appletInstaller$` directory on your server and update the `$DocumentRoot/appletInstaller$/filesList.props` file to include the name of the options file. In this case, from `optionsFile.txt` to the end of every `.files` line. Then add the following options to all the `downloads.execute` lines: `-silent -options optionsFile.txt`.

   Note: When running a silent install, the applet will appear during the download of the files from the server, but after the InstallShield installer is started the user will not receive any status information about the installation process or its completion.
7. Launch a web browser from the client desktop PC, and point it to the `$host$/appletInstaller` directory to launch the installation - where $host$ is replaced with `http://<your_http_server>`. For the example we have been using, the URL would be `http://<HTTP_SERVER>/appletInstaller`.

If your browser supports Java then the applet should be loaded. You will be prompted to accept the test certificate provided. Click **yes**, and the applet begins downloading the client image appropriate for your client device. Once the applet is complete, it launches the InstallShield installer and you can then follow the instructions in the *WebSphere Everyplace Deployment for Windows and Linux Getting Started Guide* to complete the installation.

If your browser does not support Java, then you will not be prompted to accept a certificate, rather, the browser will begin the download of the zip file you created containing all the necessary installation files for your client platform. It will also ask you to either open or save the files to your local system. Extract the files from the zip file into a temporary location, and then manually run the `setupwin32.exe` file or `setuplinux.bin` file from the temporary directory as appropriate for the client platform.

If you would like to make any modifications to the appletInstaller source code, please refer to the section, "Modifying the appletInstaller sample source code" of the `setup.htm` file in the `/install/applet` directory on your installation media.

For more information on customizing the files to be deployed, such as automatically deploying a new application during the initial install, please see "Customizing setup type components."

Adding a new download file for a different location

Add the download file name in the `filesList.props` file, and modify the `downloadApplet.js` script to support the platform by adding the platform and its browsers to the BROWSERS Array and OS Array. The applet code will need to change only if special handling is needed on the client side to launch the downloaded application.

Changing the branding

The applet will look for a `$DocumentRoot/appletInstaller$/`*`resources`* directory on the HTTP server to locate branding information. If the branding files cannot be found, the applet uses the resources packaged in the Installer jar. The resources directory will serve as an override to the images and strings that are used to brand the applet. The only restriction is that no new strings can be added, and the image sizes are the same as the default images provided. If you would like to add your own branding images, create a directory called `$DocumentRoot/appletInstaller$/`*`resources`* directory on the HTTP server and place the files from the following directory into it:

```
<installation_media>/install/applet/src/com/ibm/we/install/applet/resources/
```

These image names and sizes can then be used as a basis for creating new branding for the appletInstaller.

Customizing setup type components

The WebSphere Everyplace Deployment installation process consists of two steps: installation of core platform files and provisioning of additional components. Depending on the type of setup desired (service, minimal, full), differing sets of components are provisioned to the product from update sites.

The set of components to be provisioned are defined within provisioning configuration files, which are processed by the installer and platform provisioning system. These files must be present in a `deploy/` directory located in the same location as the installer executable. The configuration files must be named `install_`*`<setup type value>`*`.properties`.

The WebSphere Everyplace Deployment installation process supports two user-selectable setup type values: minimal (value=1), and full (value=2). A third value is available for installing as a system management service, (value=0). The WebSphere Everyplace Deployment CD includes a configuration file for each of these setup types. You cannot modify these files because they reside on the CD. However, if the contents of the `install/` directory were copied to a read-write file system (as in the case of a Web download scenario), then the files can be modified to have each of these setup types include a customized set of components.

Note: Modifying these configuration files from their shipped states could result in an invalid platform installation.

During the installation process, the configuration file corresponding to the chosen setup type is processed (replacing any supported variables) and copied to `<product destination>/rcp/deploy/install.properties.` The location of this file is then passed to the platform provisioning component to begin the provisioning process. When complete, the provisioning component will rename the provisioning configuration file to `install.properties.bak` to denote that the file has been processed.

Configuration file

The configuration file is a standard Java properties file that contains `key=value` pairs. The format of the configuration file is as follows:

```
config.size=<optional - total installed size in bytes of features listed in this configuration>
config.size.nl1=<optional - total installed size in bytes of Group 1 Localization packs included by features listed in this configuration>
config.size.nl2=<optional - total installed size in bytes of Group 2 Localization packs included by features listed in this configuration>
siteX.url=<site URL>
siteX.name=<optional - site name>
siteX.featureY.id=<feature ID>
siteX.featureY.version=<feature version>
```

You can specify multiple sites in a configuration file, and you can specify multiple features from each site. You must give each site a URL. URLs must use either the `file:` or `http:` protocol. Sites that are located relative to the path of the installer executable can be prefixed with the INSTALLER_HOME variable, which will be replaced with the correct path during the WebSphere Everyplace Deployment installation. Configuration install sizes can also be optionally specified. These sizes will be added to the size of the files laid down by the installer so that an accurate total platform size (including installed and provisioned files) can be presented to the user.

In listing sites and features, each property must have a unique name. The sites and features are numbered (starting at 1) and increment for each addition site or feature. If you have three sites defined in the file, they should be numbered site1, site2, site3. The numbering of sites and features must be contiguous (for example, must be site1.feature1, site1.feature2, and site1.feature3, not site1.feature1, site1.feature2, site1.feature5), or the additional sites and features will be ignored. These properties can be arranged in any order within the file.

Examples

1. Installation of a component located on a CD-ROM:
```
config.size=50000
site1.url=INSTALLER_HOME/site
site1.name=CD-ROM site
site1.feature1.id=com.ibm.pvc.samples.orderentry.feature
site1.feature1.version=6.0.0
```
2. Installation of a component located on a file system:
```
site1.url=file:/c:/site
site1.name=file system site
site1.feature1.id=com.ibm.pvc.txncontainer.feature
site1.feature1.version=6.0.0
```
3. Installation of a component located on two different HTTP server sites:
```
site1.url=http://www.ibm.com/site1
site1.name=HTTP Site 1
site1.feature1.id=com.ibm.pvc.samples.orderentry.feature
site1.feature1.version=6.0.0
site2.url=http://www.ibm.com/site2
site2.name=HTTP Site 2
site2.feature1.id=com.ibm.pvc.txncontainer.feature
site2.feature1.version-6.0.0
```

Uninstalling WebSphere Everyplace Deployment

Use the steps in this section to uninstall WebSphere Everyplace Deployment.

1. Launch the WebSphere Everyplace Deployment 6.0 **Uninstall** shortcut from the **Start** menu.

   Note: On a Windows machine, you can use the following set of alternate steps to launch the uninstall program:

   - Launch **Add/Remove Programs** by selecting **Settings** → **Control Panel** → **Add/Remove Programs**
   - Select IBM WebSphere Everyplace Deployment 6.0 and click **Change/Remove**.
2. Follow the prompts to remove the application.

Managing client runtimes

This section describes the tasks you can perform to configure the WebSphere Everyplace Deployment client on the user's machine.

Understanding the client file layout

When WebSphere Everyplace Deployment is installed on a machine, the installer creates a directory structure in the installation directory. This section describes the layout of the installation directory installed on the client platform.

```
<installation directory>/
    _uninst/              - Files required for uninstalling the product
    eclipse/              - Platform components
         .eclipseproduct
         configuration
         features/
         links/
         plugins/
    rcp/                  - Platform components
          eclipse/
             .eclipseproduct
             features/
             plugins/
    shared/               -Site to contain applications shared across multiple configurations
         eclipse/
             features/
             plugins/
    license/              - Product licenses in multiple languages
    migration/            - WCTME EO 5.8.1 migration utility
```

The `.eclipseproduct` marker files are used by the `installupdate` plug-in. When an installation site is marked with `.eclipseproduct` marker it is filtered so that a user cannot uninstall features installed here. A user is allowed to install into these directories, but only after a warning. An ISV should add the `.eclipseproduct` marker to their installation directories to obtain the same filtering.

You can also control how these features are updated by adding these base features to your features. The root features has the ability to control the update site that is used. See <includes> <search-location> in the feature manifest documentation for more information.

Related tasks

"Configuring the platform launcher" on page 37
You can configure additional arguments to use when the user launches WebSphere Everyplace Deployment. For example, you can specify that a console is displayed when launching the client.

"Updating the rcpinstall.properties file" on page 28
This section describes how you can update one or more elements of a user's WebSphere Everyplace Deployment client platform by modifying the `rcpinstall.properties` file. The installer creates and populates the `rcpinstall.properties` file. Users should not make changes to this file. Only ISVs and developers should make changes to the file, however, they should not change the installed values.

Related information

"Specifying platform branding" on page 32
You can modify the user interface of the client workbench to include your own branding. Use the instructions in this section to modify such elements as the title bar, splash screen, icons and images, and the About dialog.

Configuring Java system properties

Applications might require specific system properties to be set at startup when running the WebSphere Everyplace Deployment platform. To minimize the number of parameters that you must specify on the command line, you can add configuration lines to the `rcpinstall.properties` file, which resides in the `<installation directory>/rcp/` directory.

Refer to "Updating the rcpinstall.properties file" on page 28 for detailed information.

You can also specify properties on the command line when you launch the platform. Refer to "Configuring the platform launcher" on page 37 for more information.

Configuring Java VM arguments

Applications might require the addition of JVM specific arguments when the platform starts. To minimize the number of parameters that must be specified on the command line, you can add `vmarg.*` configuration lines to the `<installation directory>/rcp/rcpinstall.properties` file.

Refer to "Updating the rcpinstall.properties file" on page 28 for detailed information.

You can also specify VM arguments on the command line when you launch the platform. Refer to "Configuring the platform launcher" on page 37 for more information.

Configuring native library references

The recommended approach is that all native library objects be included in operating system/processor specific fragments. In general, this is sufficient to allow the application code and the operating system to locate the desired library. However, there might be cases where it is not possible to organize the libraries within a fragment, or the library loading requirements inhibit this approach. Therefore, library search path must be updated.

You will need to update the `library.path.append` or `library.path.prepend` lines in the `rcpinstall.properties` file to specify the directory locations containing the libraries.

Refer to "Updating the rcpinstall.properties file" on page 28 for detailed information.

Configuring Java bootclasspath libraries

The recommended approach is that all class libraries that are needed by applications reside within plug-ins or fragments. If there are cases in which the libraries must be placed on the Java bootclasspath, then you will need to update the `–Xbootclasspath.append` or `Xbootclasspath.prepend` line or lines in the `rcpinstall.properties` file.

Refer to "Updating the rcpinstall.properties file" on page 28 for detailed information.

Configuring workbench options

The default WebSphere Everyplace workbench provides configuration options that control the display of the Banner Bar, Cool Bar, and Status Bar in the user interface. You can modify the display options for each bar.

To enable or disable the display of these objects in the user interface, you will need to edit the `plugin_customization.ini` file, which resides in the following directory:

```
<installation_directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>
```

Use a text editor to open and edit this file. Change the values for `bannerVisible`, `coolBarVisible`, and `statusLineVisible` to true or false as appropriate.

Setting **bannerVisible=true** and **coolBarVisible=true** provides the capability to display these areas. The user will also have options to display or hide these areas. If these are set to false, they will not be displayed and the user will not be able to change the settings

`statusLineVisible` either hides or displays the status bar. If this is set to true, it will always be displayed, and the user will not be able to hide this area.

The other content in this file controls the appearance of the banner that is displayed in the Banner Bar. Refer to "Specifying platform branding" on page 32 for more information.

Configuring the web container

The default configuration for the web container listens only for HTTP requests received on `localhost` on a port dynamically selected during platform startup. If you need to make changes to this configuration, refer to the contents of this section.

The following properties are available for configuration of the web container.

Table 1. Java system properties

Option	Default	Description	Java System Property	ConfigurationAdmin key
HTTP Port	0	Defines the port used by the HTTP Service listener to listen for requests. A value of 0 indicates that a port will be selected at random when the platform starts. A value of -1 indicates that no listener will be started to listen for HTTP requests.	com.ibm.pvc.webcontainer.port or com.ibm.osg.webcontainer.port	http.port

Table 1. Java system properties (continued)

Option	Default	Description	Java System Property	ConfigurationAdmin key
HTTPS Port	-1	Defines the port used by the HTTPS Service listener to listen for requests. A value of 0 indicates that a port will be selected at random when the platform starts. A value of -1 indicates that no listener will be started to listen for HTTP requests.	com.ibm.pvc.webcontainer.port.secure or com.ibm.osg.webcontainer.port.secure	https.port
HTTP Timeout	60 sec	Defines the value used for socket time-outs	com.ibm.pvc.webcontainer.http.timeout or com.ibm.osg.webcontainer.http.timeout	http.timeout
HTTP Address	localhost	Defines the host address for the default ports that the Web Container listens on. If this property is defined then the Web Container will only listen for requests that come through this IP address. The special value ALL indicates all available IP addresses on the device will be used. The value of this property may be a resolved name or IP address (e.g. www.ibm.com, 192.168.0.101, localhost).	com.ibm.pvc.webcontainer.http.address or com.ibm.osg.webcontainer.http.address	http.address
Min HTTP Threads	4	Defines the minimum number of threads to be started to listen for requests.	com.ibm.pvc.webcontainer.http.minThreads	http.minThreads
Max HTTP Threads	20	Defines the maximum number of threads to be started to listen for requests.	com.ibm.pvc.webcontainer.http.maxThreads	http.maxThreads
Max Keep Alive Connections	20	Use this property to specify the maximum number of concurrent keep alive (persistent) connections across all HTTP transports.	com.ibm.pvc.webcontainer.http.maxKeepAliveConnections	http.maxKeepAliveConnections
Max Keep Alive Requests	50	Use this property to specify the maximum number of requests that can be processed on a single keep alive connection.	com.ibm.pvc.webcontainer.http.maxKeepAliveRequests	http.maxKeepAliveRequests
Keep Alive Timeout	20 sec	Use this property to specify the maximum number of seconds to wait for the next request on a keep alive connection.	com.ibm.pvc.webcontainer.http.keepAliveTimout	http.keepAliveTimeout
MIME Mapping configuration file	mimemap.properties	Specifies the MIME mapping file to be used by the Web Container. By default, the Web Container will use the mimemap.properties file located in the Web Container plug-in directory.	com.ibm.pvc.webcontainer.mimemap.configfile	n/a - cannot be configured with Configuration Admin.
Encoding configuration file	encoding.properties	Specifies the character set encoding file to be used by the Web Container. By default the Web Container will use the encoding.properties file located in the Web Container plug-in directory. Users can use this property to supply their own character set encoding file. Users can specify an absolute path or a path relative to the working directory.	com.ibm.pvc.webcontainer.encoding.configfile	n/a - cannot be configured with Configuration Admin.

If your machine is disconnected from the network (Linux only) then you need to manually add the following entry to the `/etc/hosts` file:

```
127.0.0.1 <your_machine_hostname>
```

This will allow the Web Container to function correctly when you are disconnected from the network

Configuring the web container to use a dynamic port

If the value of either the `com.ibm.pvc.webcontainer` or `com.ibm.osg.webcontainer` Java system properties is equal to 0, then the web container selects a random port when the web container plug-in is started. This allows for multiple instances of the Web Container to be running at the same time on the same machine.

Note: This port will be broadcast to all plug-ins that register an HttpSettingListener service.

Configuring with system properties

For simple configuration changes, you can effect the changes by adding Java system properties to the platform. For each of the configuration properties identified in "Configuring the web container" on page 25, add the appropriate Java system property to the `rcpinstall.properties` file. Refer to "Configuring Java system properties" on page 24 for information on adding Java system properties.

Note: If any of the system properties are specified, then the system properties will take precedence and the Web Container will ignore configuration data stored in the ConfigurationAdmin.

Configuring with Configuration Admin

The Web Container is configurable using the OSGi Configuration Admin service. The Enterprise Management Agent provides a means for administrators to configure the Web Container using the Configuration Admin service.

Note: If any of the system properties are specified, then the system properties will take precedence and the Web Container will ignore configuration data stored in the Configuration Admin.

The PID of the Web Container `ManagedServiceFactory` is `com.ibm.pvc.webcontainer`.

Refer to "Configuring client plug-ins with the Enterprise Management Agent" on page 52 for more information on how to configure the Web Container using Configuration Admin and the Enterprise Management Agent.

Using the Admin Utility for OSGi to access ConfigurationAdmin

Applications can configure the Web Container using the Configuration Admin service API. You can also configure these parameters using the Admin Utility for OSGi. To configure the Web Container using the Admin Utility for OSGi, perform the following steps:

1. Launch the WebSphere Everyplace Deployment platform.
2. Install the Admin Utility for OSGi feature. The Admin Utility for OSGi feature is located on the installation media in the `updates/eval` update site.
3. Select **Application Open > Admin Utility for OSGi** to run the application.
4. When the application is started, select the **Web Container** from the list of installed plug-ins. The HTTP Service Settings will display a list of configurable options. You might have to scroll down to find the configurable HTTP service settings.

   Users can configure the following Web Container settings:

   - HTTP Port
   - HTTPS Port

- HTTP Timeout
- HTTP Address
- HTTP Redirect Port
- Minimum HTTP Threads
- Maximum HTTP Threads
- Maximum Keep-Alive Connections
- Maximum Keep-Alive Requests
- Maximum Keep-Alive Timeout

5. If you want to use settings other than the default, enter the new value for the setting and click **Configure**.

Configuring HTTPS

The secure hypertext transfer protocol (HTTPS) is a communications protocol used to encrypt data between computers over the internet. HTTPS uses a Secure Socket Layer (SSL) to transfer encrypted HTTP data. Refer to “Configuring SSL for the Web Container” on page 72 for the required steps.

Configuring Web services

No local configuration is needed to deploy Mobile Web services. Mobile Web services providers will listen on the same port of the Web container. Please note that if a Mobile Web services client application results in a runtime exception with the message ″Parsing of the specified WSDL failed″, followed by an explanation, you must ensure that the WSDL is accessible through a browser. This exception could either be the result of a firewall message in HTML-form requiring authentication, or could be due to an invalid WSDL. If the WSDL is valid, authenticating with the firewall prior to running the Mobile Web services client application or hosting the Web service and/or WSDL outside the firewall may be needed.

Configuring platform configuration files

Specifying other platform configuration properties

The following system properties can be specified as with any other Java system properties. These will control operation of the platform. Some of the properties are not currently set and can be specified if the platform behavior needs to be changed.

provisioning.configFile
Defines the path to the provisioning config file that lists the features to install and the sites to install from. Refer to “Customizing setup type components” on page 20 for more information.

provisioning.errorFile
Defines the file where provisioning errors will be written. The installer uses this file to log any errors that occur during provisioning. If no errors occurred, then this file is not created.

Updating the rcpinstall.properties file

This section describes how you can update one or more elements of a user’s WebSphere Everyplace Deployment client platform by modifying the `rcpinstall.properties` file. The installer creates and populates the `rcpinstall.properties` file. Users should not make changes to this file. Only ISVs and developers should make changes to the file, however, they should not change the installed values.

The `rcpinstall.properties` file resides in the same directory as the launcher, which is `<`*`installation directory`*`>/rcp`. This file must meet all requirements of a Java property file. Any non-ASCII characters must be escaped with \uxxxx. You can generate these values with your favorite Unicode 16 editor and converting with the Java program native2ascii. The `rcpinstall.properties` file can contain the following properties:

Note: Unless indicated otherwise, all properties should appear only once within the `rcpinstall.properties` file. If a property appears more than once, only the first occurrence of the property is used.

Table 2. rcpinstall.properties

Property	**Description**
vm=<JVM executable file>	REQUIRED. This must point to a JVM executable file. The preference is javaw .
rcp.install.id=<id>	REQUIRED. This is a unique id that the installer creates for each installation.
cp=<classpath>	REQUIRED. This must include <install_directory>/eclipse/startup.jar
Xbootclasspath.prepend=<path>	Specifies the **-Xbootclasspath/p**:*path* See the documentation for the Java application launcher.
Xbootclasspath.append=<path>	Specifies the **-Xbootclasspath/a**:*path* See the documentation for the Java application launcher.
-D<prop>=<value>	See the documentation for the Java application launcher. Additional System properties may be added at the bottom of the file.
library.path.append=<path>	Modifies PATH (on Windows) or LD_LIBRARY_PATH (on Linux)
library.path.prepend=<path>	Modifies PATH (on Windows) or LD_LIBRARY_PATH (on Linux)
application=<application plugin id>	REQUIRED. This is equivalent to the eclipse runtime property `eclipse.application`. You can override this for the user and serial.multiuser configurations by using the `-application` argument on the command line.
install.configuration={serial.multiuser, service, user}	REQUIRED. Specifies whether the installation is a service, serial multi-user, or a user. After a platform has been installed, this value should not be changed.
vmarg.<name>=<value>	You can provide VM specific arguments here. The <value> is passed unaltered as a vmarg to the JVM. The <name> is only used as a unique identifier of the vmarg within this file. Syntax of the argument is not checked. You can add additional VM arguments to the bottom of the file.

Table 2. rcpinstall.properties (continued)

Property	Description
library.preload=<value>	REQUIRED (Linux only). This will have been populated by the installer if needed. LD_PRELOAD is reset to this value.

An example of a typical `rcpinstall.properties` file might look like the following:

In this example, lines have been split to enable readability. In the actual file, the lines must not be split unless '\' is used. The line continuation character ('\') may be used with the following restrictions related to <prop>=<value>:

1. It must be the last character on the line. There can be no trailing white space.
2. If the <value> is split, do not insert white space before the continuation character.
3. Any white space at the beginning of a continued line is discarded.
4.

For the `library.path.append` property, you can split as follows:

```
library.path.append=;\
   C\:/Program Files/IBM/..../InventoryScanner;\
C\:/Program Files/IBM/..../os/win32/x86;\
C\:/Program Files/IBM/..../os/win32/x86;\
C\:/Program Files/IBM/..../InventoryScanner
```

```
#Fri Jun 03 10:22:45 CDT 2005
install.configuration=user
vm=C\:/Program Files/IBM/WEDMGMT/rcp/eclipse/plugins/com.ibm.rcp.j2se.win32.x86_1.4.2.SR1/jre/bin/javaw.exe
library.path.append=;
C\:/Program Files/IBM/WEDMGMT/rcp/eclipse/plugins/com.ibm.tivoli.agentext.win32.x86_1.8.0.20050601/InventoryScanner;
C\:/Program Files/IBM/WEDMGMT/rcp/eclipse/plugins/com.ibm.db2e.win32.x86_8.2.1-20050601/os/win32/x86;
C\:/Program Files/IBM/WEDMGMT/rcp/eclipse/plugins/com.ibm.mobileservices.isync.win32.x86_8.2.1-20050601/os/win32/x86;
C\:/Program Files/IBM/WEDMGMT/rcp/eclipse/plugins/com.ibm.tivoli.agentext.win32.x86_1.8.0.20050601/InventoryScanner
-Dprovisioning.errorFile=C\:/Program Files/IBM/WEDMGMT/rcp/deploy/error.log
rcp.installId=1117812134309
Xbootclasspath.append=C\:/Program Files/IBM/WEDMGMT/rcp/loggerboot.jar
rcp.install.id=1117812134309
vmarg.com.ibm.jre.Xj9=-Xj9
-Dprovisioning.configFile=C\:/Program Files/IBM/WEDMGMT/rcp/deploy/install.properties
application=com.ibm.eswe.workbench.WctWorkbenchApplication
cp=C\:/Program Files/IBM/WEDMGMT/eclipse/startup.jar
-Djava.util.logging.config.class=com.ibm.rcp.core.logger.boot.LoggerConfig
-Dcom.ibm.osg.service.osgiagent.logfileloc=C\:/Program Files/IBM/WEDMGMT/rcp
-Dcom.ibm.osg.service.deviceagent.nativeinstall.default=C\:/Program Files/IBM/WEDMGMT/shared/eclipse
```

Updating the config.ini file

The `config.ini` file is located in the `<`*`installation directory`*`>/eclipse/configuration` directory. The installer provides information and values for this file that the platform needs. The remaining configuration files are located in the `.config` directories. You can add a `config.ini` file to the `.config` directory. Values in the `.config/config.ini` file will override the values in the global `eclipse/configuration/config.ini` file.

The location of the `.config` directory is dependent upon the installation configuration you chose:

If the installation configuration is **service**, then the location of the `.config` directory is: `<`*`installation directory`*`>/eclipse/.config/`

If the installation configuration is **serial multiuser**, then the location is: `<`*`installation directory`*`>/eclipse/.config/`

If the installation configuration is **user**, then the location is:
`<user.home>/IBM/RCP/<rcp.install.id>/<user.name>/.config/`

<user.home> and <user.name> are Java system properties.

<rcp.install.id> is from the rcpinstall.properties file.

The following is a list of properties found in the config.ini file:

osgi.framework.extensions
Extensions to the base Eclipse framework.

Current Setting: `com.ibm.jxesupport`

Default setting: `<none>`

osgi.parentClassloader
The definition of the parent class loader for OSGi bundles.

Current Setting: `ext`

Default setting: `boot`

osgi.splashPath

The comma separated list of URLs to search for the file named `splash.bmp`.

Current Setting:

```
platform:/base/../rcp/eclipse/plugins/com.ibm.pvc.wct.platform,
platform:/base/../rcp/eclipse/plugins/com.ibm.pvc.wct.platform.nl1,
platform:/base/../rcp/eclipse/plugins/com.ibm.pvc.wct.platform.nl2
```

Default Setting:`<none>`

osgi.bundles

A comma separated list of bundles that will be installed and optionally started once the system is up and running. For more information on the syntax for this property, refer to the Platform Plug-in Developer's Guide.

Current Setting:

```
org.eclipse.core.runtime@2:start,
org.eclipse.update.configurator@3:start
com.ibm.pvc.wct.platform.autostart@3:start
```

Default Setting: `<none>`

osgi.bundles.defaultStartLevel

Assigns the default start level to any bundles that do not explicitly have a start level assigned.

Current Setting: `4`

Default Setting: `4`

eclipse.exitOnError

Indicates whether the workbench should exit immediately if it receives a Framework ERROR event.

Current Setting: `false`

Default Setting: `true`

eclipse.product

Sets the identifier of the product being run, which identifies the branding information associated with the workbench.

Current Setting: `com.ibm.pvc.wct.platform.WctWorkbenchProduct`

Default Setting: `<none>`

java.protocol.handler.pkgs
Java System property that specifies classes that will be used to handle different URL types.

Current Setting: `com.ibm.net.ssl.www.protocol`

Default Setting: `<not set>`

Specifying platform branding

You can modify the user interface of the client workbench to include your own branding. Use the instructions in this section to modify such elements as the title bar, splash screen, icons and images, and the About dialog.

Note: In this section we indicate the changes that need to be made relative to the installed `com.ibm.pvc.wct.platform` plug-in and its files. Refer to "Distributing branding updates" on page 37 for additional considerations on how to make these changes.

The following diagram depicts the location of various objects that you can modify when specifying platform branding.

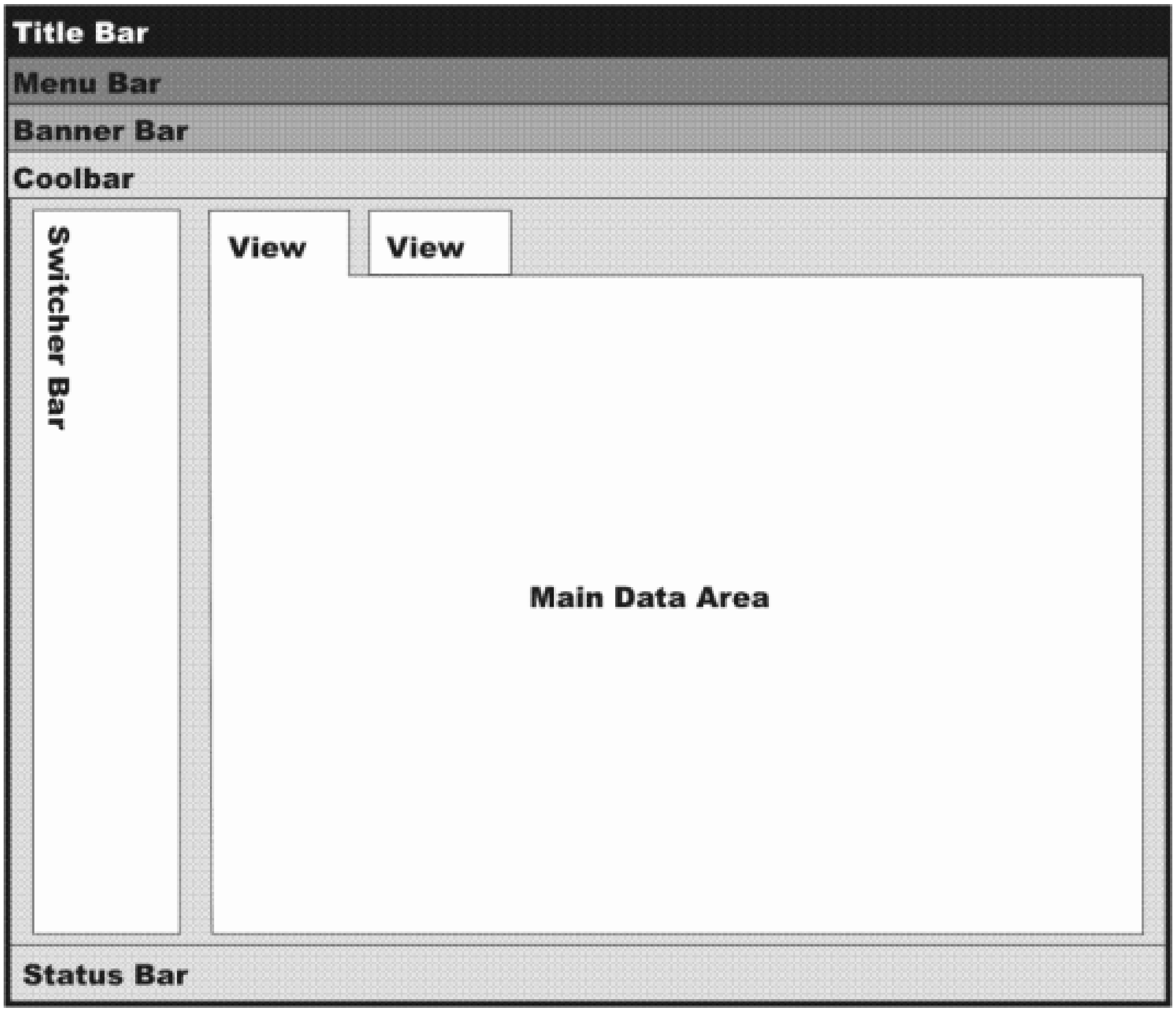

Figure 3. Configurable objects in the workbench

Changing the title bar

The title bar appears at the top of the client platform and usually contains the name of the workbench, and a small graphic.

To modify the product title bar, you modify the name attribute of the product extension in the `plugin.properties` file. The `plugin.properties` file resides in the following directory:

`<`*`installation directory`*`>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<`*`version_no`*`>`

The following is an example of the product extension in the`plugin.properties` file. To rename the title bar, you can specify the new name in the `product name` attribute:

```
<extension
 id="WctWorkbenchProduct"
 point="org.eclipse.core.runtime.products">
 <product
  name="%product.name"
        application ...
```

Changing the splash screen

When a user launches the workbench, a splash screen image is displayed. You can replace the splash screen image with your own image. The splash screen image is a file called `splash.bmp`. The splash screen must have the file extension .bmp. There are no constraints regarding the size of the image, but for reference, the standard Eclipse splash screen image is 500 x 300 pixels.

You can have a different splash screen for each locale that the product supports. When the application starts, the launcher determines the locale of the machine, and then selects the splash screen image from the appropriate language directory in the `plugins` directory. For example, a splash screen for the French locale would reside in the `nl/fr` directory.

To replace the splash screen image, replace the `spash.bmp` file that resides in the following directory:

`<`*`installation directory`*`>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform.nl1_<`*`version_no`*`>/nl/<`*`locale`*`>`

If the launcher does not find a splash screen image for your locale, then the launcher selects the default image from the `<`*`installation directory`*`>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<`*`version_no`*`>` directory.

Changing the product images

On Windows systems, a small image is associated with the product and is displayed in the title bar, next to the product title. You can modify this image to be consistent with your branding.

Standard Widget Tools (SWT) allows a set of images to be associated with a shell with the expectation that all the images in the set will have the same appearance but be rendered at different sizes. These images are provided to the SWT shell, which is then able to select the most appropriate one for each specific use. For example, the smaller image (16 X 16) is used for the title and task bars while the larger image (32 X 32) is used in the Alt-Tab application switcher.

To modify the images, replace images **16.gif**, **32.gif** and **48.gif** files, which reside in the `<`*`installation directory`*`>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform.nl1_<`*`version_no`*`>` directory.

Changing the About dialog

To specify branding of the About dialog, you must replace the image shown in the About dialog box, and also the text that displays next to the image.

Changing the About dialog image:

The image shown in the About dialog is supplied by a file called `about.bmp`. You can replace this file with the image you want to use, but it must be a file with a **.bmp** extension.

You can have a different About dialog images for each locale that the product supports. When a user opens the About dialog, the system detects the locale of the machine, and then selects the About dialog image from the appropriate language directory in the `plugins` directory. For example, a splash screen for the French locale would reside in the `nl/fr` directory.

To replace the About dialog image, replace the `about.bmp` file in the following directory:

```
<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform.nl1_<version_no>/nl//<locale>
```

.

If the system does not find a About dialog image for your locale, then the system selects the default image from the following directory:

```
<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>
```

Changing the About dialog text:

The text that is displayed next to the image in the About dialog is contained in the `plugin.properties` file. The `plugin.properties` file is locale-specific. For each locale, you can modify the `productAboutText` property in the `plugin.properties` file, which resides in the following directory:

```
<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>/nl/<locale>
```

If there is no`plugin.properties` file for your locale, then you can modify the default `plugin.properties` file, which resides in the `<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>` directory.

Changing the background image

The background image is the image that displays in the main data area of the workbench when no applications are opened. You can modify the background to display your own image.

To display your own image, replace the `default_background.gif` file in the following directory:

```
<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>/nl/<locale>
```

If there is no `default_background.gif` file for your locale, then you can modify the `default_background.gif` file, which resides in the `<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>` directory.

You can also change the image path and file name by modifying the `defaultBackgroundImage` property in the `plugin_customization.ini` file, which resides in the `<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>` directory.

To point to an image in the `nl` directory, you can set the value of `defaultBackgroundImage` to `$nl$/new_default_background.gif`.

Changing the Switcher Bar

You can change the image that is displayed in the switcher bar to an image of your choice. The image that you provide will need to be tiled to fill in the full area of the switcher bar.

To display your own image, replace the `switcherbar_background.gif` file in the following directory:

```
<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>/nl/<locale>
```

If there is no `switcherbar_background.gif` file for your locale, then you can modify the `switcherbar_background.gif` file, which resides in the`<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>` directory.

You can also change the image path and file name by modifying the `switcherBarImage` property in the `plugin_customization.ini` file, which resides in the `<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>` directory.

To point to an image in the **nl** directory, you can set the value of `switcherBarImage` to `$nl$/new_switcherbar_background.gif`.

Changing the Banner bar

The banner area resides at the top of the workbench window, directly below the menu bar. You can customize all of the visual elements of the banner to suit your needs. You can change the default height of the banner, or hide the banner in environments where screen real estate are a concern.

You modify the banner area by setting properties in the `plugin_customization.ini` file, which resides in the `<installation directory>/rcp/eclipse/plugins/com.ibm.pvc.wct.platform_<version_no>` directory.

The banner area consists of an image on the right, an image on the left, and a tiled background image. Switching between applications will change the name and images displayed in the banner area. The system derives the application name to use from the `WctWebApplication` and `WctApplication` extension points within each application.

The following table lists all of the properties that you can configure for your banner in the `plugin_customization.ini` file:

Table 3. plugin_customization.ini properties

Preference	Type	Default	Description
bannerVisible	Boolean	true	Determines whether or not the banner is visible.
bannerHeight	Integer	60	Determines the height of the banner area in pixels
bannerBackgroundColor	n:n:n where 0 <=n <=255	0:0:0	RGB value for the banner background. Visible if no bannerTileImage is specified.

Table 3. plugin_customization.ini properties (continued)

Preference	Type	Default	Description
bannerTileImage	string		Filename of the horizontally tiled background image.
bannerRightImage	string		Filename of the right-aligned image on the banner.
bannerLeftImage	string		Filename of the left-aligned image on the banner.
applicationTitleVisible	Boolean	true	Determines whether the application title is displayed.
applicationTileXpos	integer	61	x offset in pixels for the page title (measured from the left).
applicationTitleYpos	n | centered where n=0	5	y offset in pixels for the page title (measured from the top)
applicationTitleColor	n:n:n where 0 <=n <=255	255:255:255	RGB value for the page title text.
applicationTitleFontSize	integer	14	Size of the page title text in points.

Distributing branding updates

While each of the changes discussed here can be made to the actual files located on disk, you will most likely want to distribute these changes to all clients that you are responsible for. If you want to make branding changes, it is recommended that you make the changes prior to distributing the client. The recommended approach for making these changes is the following:

- Make a copy of the `com.ibm.pvc.wct.platform` plug-in, and any language fragments of that plug-in that you require, creating a new symbolic name for the plug-in (and updating the host plug-in for each of the fragments)
- Follow the instructions above in order to make any branding changes that you require
- Create a feature to include your new plug-in
- Update the `rcpinstall.properties` file to remove the `com.ibm.pvc.wct.platform` feature and replace it with your new feature. Refer to “Customizing setup type components” on page 20 for more information.
- Provide access to the updated distribution site for your users.

Configuring the platform launcher

You can configure additional arguments to use when the user launches WebSphere Everyplace Deployment. For example, you can specify that a console is displayed when launching the client.

The options that you might want to configure are:

```
-console
-debug
-consoleLog
-application
```

```
-vmargs
-configuration
-data
-nosplash
-rcpLauncherWait
```

`-console` enables you to display the OSGi console. You can have the maximum information logged by adding `logredirector.level=INFO` to the `rcpinstall.properties` file. These two options work well together when you need to debug problems.

`-consoleLog` will cause every console message to be duplicated. It is better to use the -console option instead.

`-application` temporarily overrides the `rcpinstall.properties` file.

`-configuration` temporarily override the `osgi.configuration.area` property, which is calculated using the`install.configuration` property in the `rcpinstall.properties` file.

`-data` temporarily overrides the `osgi.instance.area` property, which is calculated using the `install.configuration` property in the `rcpinstall.properties` file.

`-rcpLauncherWait` will cause the parent launcher to wait until the launch is complete. Normally, this parent does not wait. The installer uses this option to determine when the launch is complete.

Arguments based on Java VM, Eclipse, or OSGi are passed through, except for the following:

- The `-vm` property is required in the `rcpinstall.properties` file. When the -vm argument is passed to the `rcplauncher.exe` the argument is ignored.
- If either the `-console` or `-consoleLog` argument is specified, then `javaw.exe` specified in the `rcpinstall.properties` file will be converted to `java.exe` by `<launcher>.exe`.
- If the platform is running as a service the `-nosplash` option is always active.

For a complete list of runtime options, refer to the Platform Plug-in Developer's Guide, installed with the Rational Software Development Platform.

To configure the platform launcher to use additional arguments, you must add the argument to the path of the executable. To add arguments to the executable, you modify the shortcut from the desktop icon, or if the WebSphere Everyplace Deployment client is installed as a Windows service, you modify the service's image path from the registry. If the client is installed as a Linux service, you modify the /etc/init.d/mgmtservice script.

Note: The following procedure for enabling an interactive console for a Windows service should only be performed as a debugging aid. Enabling the console may introduce security holes. This makes it possible for another application running on the desktop to interact with the service. Also note that the interactive console will not work with fast-user switching, Remote Desktop, or Terminal Services. This option can only be utilized by an administrator logging in locally to the target machine.

The following example modifies the WebSphere Everyplace Deployment Client that is running as a Windows service to display a console during launch. When you follow this example, you enable additional debugging information and provide access to the OSGi console. Within the OSGi console at the osgi> prompt, type "`help`" to see a list of possible commands:

1. From the **Start** menu select **Run**.
2. In the Run dialog, type `regedit`. The registry is displayed.
3. Select:

   `HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/WEDMgmtService`

   .
4. Double click on the entry named **ImagePath**. The Edit String dialog is displayed.
5. Type `-console` at the end of the path and click **OK**.
6. Open the Services Control Panel.
7. Open the properties for WED Management Service.
8. Click the **Log on** tab and select the **Allow service to interact with desktop** option.
9. Start the service.

For Linux, you can get access to the OSGi console and additional debugging messages using the following steps:

1. Stop the service if it is running.
2. From a command line interface, type `mgmtservice -debug`. The arguments `-console` and `-consoleLog` will be added automatically when you use the `-debug` argument.
3. When finished, type exit at the `osgi>` prompt and the service will terminate. You can also type `mgmtserstop` at a different command prompt to stop the service

   Related tasks

   "Updating the rcpinstall.properties file" on page 28
   This section describes how you can update one or more elements of a user's WebSphere Everyplace Deployment client platform by modifying the `rcpinstall.properties` file. The installer creates and populates the `rcpinstall.properties` file. Users should not make changes to this file. Only ISVs and developers should make changes to the file, however, they should not change the installed values.

   Related information

   "Understanding the client file layout" on page 23
   When WebSphere Everyplace Deployment is installed on a machine, the installer creates a directory structure in the installation directory. This section describes the layout of the installation directory installed on the client platform.

   "Specifying platform branding" on page 32
   You can modify the user interface of the client workbench to include your own branding. Use the instructions in this section to modify such elements as the title bar, splash screen, icons and images, and the About dialog.

Managing client runtimes with a Device Management server

The Enterprise Management Agent enables you to connect to a WebSphere Everyplace Deployment Server, or to a WebSphere Everyplace Device Manager (WEDM) server. Both of these products contain the Device Management server component. These products provide basic platform management of the client. The WebSphere Everyplace Deployment server provides additional features, such as ISync and MQe servers.

This section describes how to use a Device Management server to install applications and manage runtimes.

Configuring the Enterprise Management Agent

The Enterprise Management Agent is provided with WebSphere Everyplace Deployment and enables you to manage the platform remotely using a Device Management Server.

The Enterprise Management Agent uses a Sample Account to connect to the server. There are three ways to create the sample account:

- At install time. For instructions on configuring the Enterprise Management Agent during installation, refer to "Installing from a CD" on page 10.
- After install time. Using the Enterprise Management Agent Servlet is described in the "Using the Enterprise Management Agent Servlet" on page 46 section. Use of the Enterprise Management Agent Servlet requires the installation of an additional feature.
- After install time. Using the Enterprise Management Agent preferences page within the WebSphere Everyplace Deployment client workbench.

This section describes how to configure the agent using the Enterprise Management Agent Preference Page. You can access the Enterprise Management Agent Preference page using the **File > Preferences > Enterprise Management Agent dialog**. The following parameters appear on the Enterprise Management Agent Preference Panel:

Table 4. Enterprise Management Agent Parameters

Parameter	Description
Enable Enterprise Management Agent	Enables management of this client via the Enterprise Management Agent. Once you enable this, you will have no control over the installation, updating, or uninstalling of features. Once enabled the Enterprise Management Agent can not be easily disabled. See the Disabling Enterprise Management Agent capabilities section for instructions.
Device User Name .	The identity that a user's machine uses to connect to a Device Management Server.
Device User Password	The password that a user's machine uses to connect to a Device Management Server
Server Password	This is optional and depends on how the Device Management Server does authentication. It is only needed if the server requires it.
HTTP/HTTPS	Choose the type of connection with the server.

Table 4. Enterprise Management Agent Parameters (continued)

Parameter	Description

Server Address	Server address consists of two parts – a URI and the path to the DM Servlet to contact for enrollment. Use one of the following to specify the Device Manager server address
	• IP address
	• Host name
	• Host name plus domain name
	
	SyncML/DM is the data synchronization protocol that the agent and server use to communicate with each other. When the server has been configured to require authorization using SyncML/DM, append the following value to the Server Address
	`dmserver/SyncMLDMServletAuthRequired`
	
	IIf SyncML/DM authorization is not required by the server, append the following value to the Server Address instead:
	`dmserver/SyncMLDMServlet`

Polling Interval	Displays the polling interval in hours and minutes. The default value is 4 hours. When the interval is set to 00:00 polling is disabled, despite the value of the Enable Poll at Start Up field. This is equivalent to disabling polling. This is not configurable via the Enterprise Management Agent preference panel. It can only be configured via the Enterprise Management Agent Servlet or Device Management Server custom command job.
Polling Start Time	Displays the polling start time in hours and minutes. The default value is 2:00 am. When the polling start and end time are the same, the agent continues to poll at each polling interval. The agent will not poll when the current time is outside the Polling start and end times. This is not configurable via the Enterprise Management Agent preference panel, It can only be configured via the Enterprise Management Agent Servlet or Device Management Server custom command job.
Polling End Time	Displays the polling end time in hours and minutes. The default value is 2:00 am. When the polling start and end time are the same, the agent continues to poll at each polling interval. The agent will not poll when the current time is outside the Polling start and end times. This is not configurable via the Enterprise Management Agent preference panel. It can only be configured via the Enterprise Management Agent Servlet or Device Management Server custom command job.
Enable Poll at Start Up	Displays whether polling is enabled at startup or not. The default value is enabled. This is not configurable via the Enterprise Management Agent preference panel. It can only be configured via the Enterprise Management Agent Servlet or Device Management Server custom command job.
Test Connection	Select this button to test that the Enterprise Management Agent can connect to the Device Management Server using the parameters as specified on this page.

Note: The WebSphere Everyplace Deployment client preference panel does not allow creation of multiple accounts. The Device Management Server provides functionality to allow a user to change the default account id using a Custom Command.

You can view agent configuration information, including the Device ID, by selecting **Help > About WebSphere Everyplace Deployment> Configuration Details**.

Once the Enterprise Management Agent is configured, the **Application > Install and Manage > application > Application Management** dialogs are disabled.

The WebSphere Everyplace Deployment client can optionally be installed as a Windows Service or Linux Daemon. This type of installation provides the Enterprise Management Agent as another method for configuring the Enterprise Management Agent. Refer to "Installing for system management" on page 12 for more information.

Changing the Enterprise Management Agent properties

You can change the agent properties for a single device or for multiple devices at a time.

These steps are performed on a system running a Device Management Server using the DM Console application provided with your Device Management Server.

Guided method

Use the guided method if you want to change properties for the agent on a single device.

The following steps describe how to change properties for the agent, making use of multiple jobs to perform this task. Because there are multiple jobs, these steps may take longer, but more information is pre-populated in the dialogs. These steps also make changes only to a single device. To apply changes to multiple devices simultaneously, use the Advanced Method listed below. These steps illustrate changing the `./OSGiAgent/PollingInterval` value.

1. Select **Devices**.
2. Select **Use New Query** and **Return anything** as your search criteria.
3. Select **OK**.
4. The DM console will show a list of enrolled devices.
5. Select your device, right click and select **Submit Job**.
6. Select **Next**.
7. Select the Job Type as **Node Discovery** (Use default settings for all the other job attributes).
8. Select **Next**.
9. Select **Add Group**.
10. Enter `./OSGiAgent` as the Target URI.
11. Enter a Search depth of **5**.
12. Select **Next**.
13. Select **OK**.
14. The job has been submitted. Select **Close**.
15. You will need to wait for the job to complete (this will depend upon the configured polling interval). Once the job has completed, select the device, then click **View Inventory...**
16. Then select **Management Tree**.
17. Select the entry with the URI: **./OSGiAgent/PollingInterval**.
18. Select **Submit Job**.
19. Select **Next**.
20. Leave the defaults as supplied in the panel, then select **Next**.

21. Enter **1** for the Command Number Field.
22. Set the Data value to **00:02**.
23. Select **Next**.
24. Select **OK**.
25. Select **Close**.
26. Select **Close**.
27. Wait for the job to complete.

Advanced method

Use this method if you want to change properties for the agent on multiple devices.

The following steps describe how to change properties for the agent, but allow you to change the properties for multiple devices. Less information is pre-populated in the dialogs.

These steps illustrate changing the `./OSGiAgent/PollingInterval` value.

1. Select **File > Submit Job... > All New...**
2. Select **OSGi** as the device class.
3. Select **Currently Enrolled** for the status of the target devices, since you are configuring agent specific properties.
4. Select **Next**.
5. Select the Job Type as **Custom Command** (use default settings for all other job attributes).
6. Select **Next**.
7. Select **Replace Command**, then click **Add Group**.
8. Enter **1** as the Command Number.
9. Enter **./OSGiAgent/PollingInterval** as the Target URI.
10. Enter **00:02** as the Data.
11. Select **Next**.
12. Select **OK**.
13. Select **Close**.

The Enterprise Management Agent can optionally be installed as a Windows Service or Linux Daemon. This type of installation provides another method for changing the Enterprise Management Agent.

Disabling Enterprise Management Agent capabilities

This section describes how to disable Enterprise Management Agent connectivity for the clients.

After you have successfully configured the Enterprise Management Agent, and it has connected to the Device Management Server, the Application Management dialogs (which you access from **Application > Install and Manage > Application Management**) are disabled. If you need to disable the agent from running, you will need to change the agent properties using custom Device Management jobs.

To disable the Enterprise Management Agent, change the value for the property name **./OSGiAgent/PollingEnabled** to **false** by following the steps described in "Changing the Enterprise Management Agent properties" on page 43.

To re-enable the Enterprise Management Agent, refer to "Configuring the Enterprise Management Agent" on page 41.

Updating client configuration using a Device Management Server

This section describes how you can use a Device Management Server to update a client's configuration.

You can use a custom command job to update the Enterprise Management Agent configuration. The Enterprise Management Agent has the following configurable parameters:

Table 5. Configurable parameters

Parameter	Description	Values
PollingEnabled (Polling enabled)	Determines if polling is enabled.	true or false (case is ignored)
PollingInterval (Polling interval)	Specifies the polling interval in hours and minutes.	HH:MM.
PollingStart (Polling start time)	Specifies the polling start time in hours and minutes.	HH:MM.
PollingEnd (Polling end time)	Specifies the polling end time in hours and minutes.	HH:MM.
TempFileLoc (Temp file location) Identifies the path for temporary files.	Identifies the path for temporary files.	A fully qualified path
PollOnStartup	Determine if the agent should poll the server when it is started.	"0" or "1".

These steps are performed on a system running the Device Management Server using the DM Console application provided with the Device Management Server. To retrieve the current configuration parameters, open the DM Console and follow these steps to illustrate the ./OSGiAgent values:

1. Select **Devices**.
2. Select **Use New Query and Return anything** as your search criteria.
3. Click**OK**.
4. The DM console will show a list of enrolled devices.
5. Select your device, right click and click **Submit Job**, then click **Next**.
6. Select the Job Type as Node Discovery (Use default settings for all the other job attributes) and click **Next**.
7. Click **Add Group** and then type `./OSGiAgent` as the target URI.
8. Enter a search depth of **5** and click **Next**.
9. Click **OK**.
10. The job has been submitted. Select **Close**.
11. You will need to wait for the job to complete (the amount of time this takes depends upon the configured polling interval). When the job has completed, select the device, then click **View Inventory**.
12. Then select **Management Tree** to view the configuration.
13. Select the entry with the URI starting with: **/OSGiAgent/** that you want to change.

14. Select **Submit Job** and click **Next**.
15. Leave the defaults as supplied in the panel, then click **Next**.
16. Enter **1** for the **Command Number** field.
17. Set the Data value you want to modify and then click **Next**.
18. Click **OK**, then click **Close** on the following two panels and wait for the job to complete.

Using the Enterprise Management Agent Servlet

The Enterprise Management Agent Servlet allows you to view and modify the configuration and control the features of the Enterprise Management Agent. You can launch it from within the WebSphere Everyplace Deployment workbench or from within a web browser. When the WebSphere Everyplace Deployment client is installed as a system management service, the Enterprise Management Agent Servlet is the only way to configure and control the Enterprise Management Agent.

Starting the Enterprise Management Agent Servlet

To start the Enterprise Management Agent Servlet from within the WebSphere Everyplace Deployment client workbench, select **Application > Open > Enterprise Management Agent Servlet**.

Alternatively, if you installed the WebSphere Everyplace Deployment client as a system management service, launch the servlet by selecting **Start > Programs > IBM > WebSphere Everyplace Deployment Service > WebSphere Everyplace Deployment 6.0.0 Service Agent Settings**.

From a browser, the following URL will launch the Enterprise Management Agent Servlet

```
http://localhost:[nn]/osgiagentservlet
```

where nn is the port number. When you installed the client as a service, you were asked to specify a port value that should be used here.

For example, if port 8082 was specified at installation, then use the following Web address:

```
    http://localhost:8082/osgiagentservlet
```

Modifying Enterprise Management Agent account information

After you have started the servlet, select **Edit Account** Configuration under Admin Functions within the navigation panel on the left side to modify the current account information.

The following list provides a brief description of the options you can modify for the account:

SyncML DM account ID
The SyncML/DM account ID, which is often the same as the server ID.

Device user name
User name for the device.

The name is used when SyncML/DM authorization or HTTP basic authentication is implemented.

Device user password
User password. The password text is displayed in clear text.

The password is used when SyncML/DM authorization is implemented.

Server password
The password is used when SyncML/DM authorization is implemented.

Address type
The type of connection to the server. Possible values are HTTP or HTTPS.

Server address
Use one of the following to specify the Device Manager server:

- IP address
- Host name
- Host name plus domain name

DM server path

SyncML/DM is the data synchronization protocol that the agent and server use to communicate with each other. When the server has been configured to require authorization using SyncML/DM, append the following value to the Server Address:

```
dmserver/SyncMLDMServletAuthRequired
```

If SyncML/DM authorization is not required by the server, append the following value to the Server Address instead:

```
SyncMLDMServlet
```

Modifying device information

After the servlet has launched, select **Device Information** under Admin Functions within the navigation panel on the left side to modify the current account information.

The following list provides a brief description of the options you can modify for the account:

Device name
A unique device name for the device. The device name is often a serial number or another unique string. The device name is displayed in the Device Information window.

The device ID is automatically generated by the Enterprise Management Agent and can not be modified. It is displayed here for information only.

Model)
The device model. The device model is displayed in the Device Information window. Also known as the device class. An example value is Win32 or Linux_x86.

Default Account
Defines the account that the agent uses to communicate with the Device Management Server.

Temp File Location
Identifies the path for temporary files used by the Enterprise Management Agent during software distribution.

Polling enabled
Toggles automatic polling of the server by the Enterprise Management Agent.

Polling interval
Specifies the polling interval in hours and minutes. When the interval is set to 00:00 polling is disabled, despite the value of the Polling enabled field. This is equivalent to disabling polling.

Polling start time
Specifies the polling start time in hours and minutes. The agent will not poll when the current time is outside the Polling start and end times.

Polling end time
Specifies the polling end time in hours and minutes. The agent will not poll when the current time is outside the Polling start and end times.

log size
Specifies the number of message entries that can be logged to the agent before it wraps. The agent log can be viewed from within the Enterprise Management Agent Servlet by selecting Agent Log under Admin Functions within the navigation pane on the left hand side.

Log Threshold
Specifies the minimum error type added to the message file. The choices and corresponding values are:

- 1 for Error
- 2 for Warning
- 3 for Info
- 4 for Debug

Connecting to the server to check for jobs

From the Agent Control Panel window, you can connect to the Device Management Server to run jobs that are in the job queue for the device.

To connect to the Device Management Server, perform the following steps:

1. From the **General Functions** list, click **Get Updates**.

   The Connect to Server window opens.
2. If there are any problems with the connection, the system tray icon will flash and when the icon is clicked a popup will appear in the bottom right of the screen providing details.
3. Optionally, you can click **Agent Events** under the **Admin Functions** in the navigation panel for a list of recent server-device actions.

   For more details on the recent server-device events, click **Agent Log** from the **Function** list. The information is for all accounts and the information is listed chronologically, not by account. To refresh the agent log, click the **Refresh Agent Log** button. .

Monitoring Enterprise Management Agent events

To view recent agent-server events, click **Agent Events** from the **Function** list. The most recent events are displayed at the top of the list. The events are for all accounts and the events are listed chronologically, not by account. To refresh the event list, click the **Refresh Event Log** button.

Note: Ensure that your browser window is wide enough to display the vertical scroll bar for the event list or adjust the Agent Events window with its horizontal scroll bar.

For more details on the recent server-device events, click **Agent Log** from the **Function** list. The details of the most recent events are displayed at the top of the list. The information is for all accounts and the information is listed chronologically, not by account. To refresh the agent log, click the **Refresh Agent Log** button.

Creating a software distribution job from the client

From the Enterprise Management Agent Servlet the device user can initiate a request for the Device Management Server to send a list of available software to the device. Upon receiving a request from the device, the Device Management Server retrieves the list and sends the list to the device.

Once the device receives the list, the device user can create a software distribution job by selecting a bundle to be distributed to the device from that list.

To create a software distribution job from a device, do the following steps from the Agent Control Panel window:

1. From the **General Functions** list, click **Available Software**.

   The Available Software window opens.

   The list of available displayed is for the current account.
2. Select a software bundle from the list of available software.
3. Click **Install Selected Software** to install that software on the device.

   After requesting software to be installed, monitor the agent's status by viewing the agent events. See "Monitoring Enterprise Management Agent events" on page 48.
4. Optionally, click **Refresh Available Software** to update the software list.

Working with Enterprise Management Agent accounts

This section describes how to select an account, create an account, edit an account's configuration, and delete an account using the Enterprise Management Agent Servlet.

Selecting an account

The current account is displayed above the **General Functions** list.

To change the current account to another account, do the following steps:

1. From the **Admin Functions** list, click **Select Account**.

   The Select Account window opens.
2. Click one of the listed accounts to make it the current account.

   The new current account is displayed above the **General Function** list.

You can now select a function to be performed for the current account from the Enterprise Management Agent Control Panel window.

Creating another account

From the Agent Control Panel window, you can create another account. For each account, you enter information for the account configuration parameters.

Account configuration parameters:

SyncML/DM account ID
The SyncML/DM account ID, which is often the same as the server ID.

Device user name
User name for the device.

The name is used when SyncML/DM authorization or HTTP basic authentication is implemented.

Device user password
User password for the device. The password text is displayed in clear text.

The password is used when SyncML/DM authorization is implemented.

Server password
Optional server password.

The password is required only when SyncML/DM authorization is implemented on the DM Server.

Address type
Select HTTP or HTTPS from the list.

Server address
Use one of the following to specify the Device Manager server:

- IP address
- Host name
- Host name plus domain name

DM server path
SyncML/DM is the data synchronization protocol that the agent and server use to communicate with each other. When the server has been configured to require authorization using SyncML/DM, append the following value to the Server Address:

```
dmserver/SyncMLDMServletAuthRequired
```

If SyncML/DM authorization is not required by the server, append the following value to the Server Address instead:

```
dmserver/SyncMLDMServlet
```

Default account
Defines the account that the agent uses to communicate with the DM Server.

To create another account, perform the following steps:

1. From the **Admin Functions** list, click **Create New Account**.

   The Create New Account window opens.
2. Enter the values for the parameters.

   Note: No validation of account parameters occurs.

3. Click the **Create New Account** button.

   The new account values are displayed with the Account Configuration Saved message.

   After you have created another account, you need to connect to the server to register the device and check for new jobs.

You can now select another function from the Enterprise Management Agent Control Panel window.

Editing the account configuration

From the Enterprise Management Agent Control Panel, you can change the account configuration by doing the following steps:

1. From the **Admin Functions** list, click **Edit Account Configuration**.

   The Edit Account Configuration window opens with information for the current account.
2. Enter the new values for any of the parameters.

   Note: No validation of account parameters occurs.
3. Click **Update Account** to save the changes.

   The new account values are displayed with the Account Configuration Saved message.

You can now select another function from the Enterprise Management Agent Control Panel window.

Deleting an account

From the Enterprise Management Agent Control Panel, you can delete an account by doing the following steps:

1. From the **Admin Functions** list, click **Select Account**.

   The Select Account window opens.
2. From the list of accounts, click the account you want to delete. To delete an account, it must be the current account.

   The current account is displayed above the **Function** list.
3. From the **Function** list, click **Edit Account Configuration**.

   The Edit Account Configuration window opens with information for the current account.

   Note: There is no confirmation when deleting an account.
4. Click the **Delete Account** button.

   An Account Deleted message is displayed.

You can now select another function from the Enterprise Management Agent Control Panel window.

Editing the Enterprise Management Agent device information

From the Enterprise Management Agent Control Panel window, you can edit some of the device information. For example, you can change the default account from the list of accounts and the location where temporary files are stored.

You can change the following information for each device:

- Default account.
- Temporary file location.
- Polling information.
- Log file size and message type threshold.

The **Device name** and **Model** fields are read-only.

To change the device information, do the following steps:

1. From the **Admin Functions** list, click **Device Information**.

   The Device Information window opens.
2. Select a new **Default account** or type a new path for the **Temp file location** field.
3. Select the **Polling interval**, **Polling start time**, **Polling end time** from the drop-down lists.
4. To enable polling, ensure the **Polling enabled** field has a check mark.
5. Type a new log file size, if desired.
6. Select a new threshold for the message types that are stored in the log file.
7. Click the **Update Device Information** button to save your changes.

   The new device information for this device and the Device Configuration Saved message is displayed.

You can now select another function from the Enterprise Management Agent Control Panel window.

Configuring client plug-ins with the Enterprise Management Agent

You can change the configuration of a client plug-in that has registered itself as a Managed Service with the framework using the Enterprise Management Agent. A Managed Service represents a client of the Configuration Admin service. Plug-ins which have registered themselves as a Managed Service (e.g., Web Container) will receive configuration update notifications from the ConfigurationAdmin service.

The Enterprise Management Agent provides administrators with a means to discover these types of plug-ins and change their configuration.

Note:

- Registration as a Managed Service does not guarantee that a plug-in can be discovered by the Enterprise Management Agent. ConfigurationAdmin must be populated with the plug-ins configuration data. Refer to the OSGi specification for more details.
- Administrators choosing to configure the client plug-in using the Enterprise Management Agent must populate the ConfigurationAdmin with the plug-ins configuration data. This can be done either programmatically or by using a tool like Admin Utility for OSGi.

The following steps describe how to change the configuration of a client plug-in, making use of multiple jobs to perform the task. These tasks should be performed on a system running a Device Management Server using the DM Console application.

1. Click **Devices**.

2. Select **Use New Query and Return anything** as your search criteria, and then click **OK**. The DM console will show a list of enrolled devices.
3. Select your device, right click and select **Submit Job**.
4. Click Next, then select the **Job Type as Node Discovery** (use the default settings for all the other job attributes).
5. Select **Next**, then select **Add Group**.
6. Type `./OSGi/BundleConfiguration/<Plugin_PID>` as the Target URL.

   where `<Plugin_PID>` is the PID of the plug-in. For example, `com.ibm.pvc.webcontainer` is the PID of the Web Container
7. Enter a search depth of 5, then click **Next**.
8. Click **OK**. The job has been submitted. Click **Close**.
9. You will need to wait for the job to complete (this will depend upon the configured polling interval). Once the job has completed, select the device, then click **View Inventory**.
10. Then select **Management Tree**.
11. Select the configuration entry you wish to configure and click **Submit Job**.
12. Click **Next**. Leave the defaults as supplied in the panel, then click **Next**.
13. Enter 1 for the **Command Number Field**.
14. Set the **Data** value for the configuration entry, then click **Next**.
15. Click **OK**.
16. Click **Close**, then click **Close** again and wait for the job to complete.

Managing applications with WebSphere Everyplace Deployment

There are three ways that you can deploy an application to be installed on the user's machine. You can deploy it locally, from a Web site, or through a Device Management Server.

The following sections describe how to prepare an application for deployment using either local or web deployment. Refer to "Managing applications with a Device Management Server" on page 63 for specific instructions on how to use the Device Management Server to deploy applications.

The install/update capability of WebSphere Everyplace Deployment depends on the use of update sites containing features. Features are groupings of one or more plug-ins. The Plug-in Development Environment (PDE) provided with the Rational suite of tools provides capabilities for application developers to construct both features and update sites.

Within this section on Managing applications, the term application is used to refer to a group of one or more features that must be installed together to provide some useful application capabilities. An update site can contain one or more applications that are available for installation.

Depending upon your management and update policy, and the artifacts that you receive from either an ISV or your application developers, you might need to perform a different set of actions.

Preparing applications for local deployment

This section describes how to prepare an application to be deployed locally.

When this document refers to local deployment, it assumes that local deployment involves the distribution of media, for example, a CD or memory key, containing an update site.

Preparing the update site:

Depending upon the application that you receive, it may be provided to you as an update site, one or more features, or a set of plug-ins.

If you receive a set of plug-ins that you need to install, then you will need to create one or more features to enable installation of the plug-ins.

When you have created a feature to wrap the plug-ins, or you have received features for the application, then you will need to create an update site to enable installation of the application into the WED platform.

After you have either created an update site, or have received an update site from your application developers or an ISV, then you will need to copy the update site directory tree to your distribution media.

Testing the update site:

After preparing the distribution media, you should test installation of the application into the platform currently distributed to your users. During the application installation, you can identify prerequisite features that also need to be installed. You can either create a new update site to contain these features, or add these features to the update site that you are currently testing with.

To test the update site, perform the following steps:

1. Start WebSphere Everyplace Deployment.
2. Select **Application** → **Install**.
3. Select **New Local Location**, and type or select the location of the update site that you have created, then click **OK**.
4. Verify that the location that you just created is selected, and then click **Next**.
5. Select the feature or features that must be installed, and then click **Next**.
6. If you agree with the license terms, select **I Accept the terms in the license agreements**, and then click **Next**.
7. Select the target installation location. For application features, you should choose the `<`*`installation directory`*`>/shared/eclipse` directory if you intend the application to be used by all users. You could also choose to install the application into your configuration location. In this case, it will only be available to you, and no other users
8. Click **Finish**.
9. If prompted to verify the feature, select **Install**.
10. When prompted, restart the platform.
11. Test the application.

After you have verified the installation and operation of the application on your test system, you are now ready to distribute the application to your users. The users can follow the same steps that you performed above to install the application. Refer to **PDE Guide** → **Getting Started** → **Update Sites** with the Rational Suite® for more information on building and using update sites.

Preparing applications for Web deployment

This section describes how to prepare an application to be deployed on a Web site.

When this document refers to web deployment, it assumes that applications are being added to an update site that exists on a WebSphere Everyplace Deployment server or other web server. Rather than constructing separate update sites for each application as might be done for a local deployment, web deployment will typically consolidate all of the applications within a single update site.

Preparing the update site

Depending upon the application that you receive, it can be provided to you as an update site, one or more features, or a set of plug-ins.

If you receive a set of plug-ins that you need to install, then you will need to create one or more features to enable installation of the plug-ins.

When you have created a feature to wrap the plug-ins, or you have received features for the application, then you will need to either create the initial web update site, or add the features to an existing update site. Because the WebSphere Everyplace Deployment CD contains an update site that contains features that might not be installed on every user's platform, copying the initial update site from the CD and updating with your new applications reduces the chances of having to locate pr-requisite features for deployment.

If you have received a prebuilt update site, then you can either create a separate update site on the web server, or merge the contents of the prebuilt update site into your already existing update site.

Testing the update site

After you have prepared the features on the update site, you should test installation of the application into the platform currently distributed to your users. During the application installation, you can identify prerequisite features that also need to be installed. You can add these features to the update site that you are currently testing with.

To test the update site, perform the following steps:

1. Start WebSphere Everyplace Deployment.
2. Select **Application** → **Install**.
3. Select **New Remote Location**, and type a descriptive site name and the URL for the update site, then click **OK**.
4. Verify that the location that you just created is selected, and then click **Next**.
5. Select the feature or features that must be installed, and then click **Next**.
6. If you Agree with the license terms, select I Accept the terms in the license agreements, and then click **Next**.
7. Select the target installation location. For application features, you should choose the `<`*`installation directory`*`>\shared\eclipse` directory if you intend the application to be used by all users. You could also choose to install the application into your configuration location. In this case, it will only be available to you, and no other users
8. Click **Finish**.
9. If prompted to verify the feature, select **Install**.
10. When prompted, restart the platform.
11. Test the application.

After you have verified the installation and operation of the application on your test system, you are now ready to distribute the application to your users. The users can follow the same steps that you performed above to install the application. Refer to **PDE Guide** → **Getting Started** → **Update Sites** with the Rational Suite for more information on building and using update sites.

Configuring enterprise definitions (JNDI)

This section describes how to deploy an application that is configured with an initial set of enterprise definitions.

WebSphere Everyplace Deployment JNDI overview

The WebSphere Everyplace Deployment client provides a simple Java object JNDI registry to support the enterprise object definition needs of web applications, Enterprise Java Bean applications, and messaging applications. The WebSphere Everyplace Deployment JNDI provider enables a local naming directory for objects running in the client platform to communicate via standard Java naming APIs. The runtime client JNDI implementation is very lightweight and does not support federation of other namespaces, rather it provides a simple hierarchical namespace for client applications. In most cases applications leveraging JNDI do not need to interact directly with JNDI Name objects and simply use String representations of the names to be bound or located. If your application needs to directly interact with JNDI CompoundName objects, please note that due to the lightweight implementation, only a restricted set of JNDI syntax properties is supported for use when creating a CompoundName object. In order to ensure the correct JNDI syntax properties are used, simply use the provided NameParser implementation from the WebSphere Everyplace Deployment JNDI provider when a CompoundName object is needed.

The WebSphere Everyplace Deployment JNDI provider can be directly accessed via the provider's `InitialContextFactory` class as follows:

```
try {
 Hashtable env = new Hashtable();
 env.put(Context.INITIAL_CONTEXT_FACTORY,"com.ibm.pvc.jndi.provider.java.InitialContextFactory");
 InitialContext context = new InitialContext(env);

} catch (NamingException e) {
 e.printStackTrace();
}
```

This JNDI provider is also registered as the default JNDI provider for the WebSphere Everyplace Deployment platform so even if no provider is specified it will still use the above `InitialContextFactory` to generate the `InitialContext` object.

The WebSphere Everyplace Deployment JNDI provider does not persist objects or their state information across platform restarts, so the platform administrator is responsible for binding the objects each time the platform starts and configuring those objects as needed before binding them into the JNDI registry. While the application itself could programmatically register the objects that it needs each time the platform starts, the WebSphere Everyplace Deployment client provides another declarative model for JNDI bindings.

Objects that need to be bound into JNDI can be declared using Eclipse extension points, to be described in detail shortly, so that when a lookup request is made for a specific object via its JNDI name the JNDI provider will locate the declarative definition, create the object and return it to the client application on-demand. This "lazy" creation of objects provides for faster platform startup and memory allocation based on actual need, rather than expected need.

This capability is based on two characteristics of the bundles/plug-ins:

1. A `plugin.xml` must exist and must provide an entry for the `com.ibm.pvc.jndi.provider.java.binding` extension point
2. If the bundle/plug-in has a `MANIFEST.MF` file, it must contain the `Eclipse-AutoStart: true` entry.

The WebSphere Everyplace Client Toolkit leverages this declarative JNDI capability to automatically generate the required `plugin.xml`, and `MANIFEST.MF` EJBs so that the WebSphere Everyplace Deployment JNDI provider can locate their declarative information upon a lookup of their JNDI name.

Using declarative JNDI

The declarative JNDI component is based on an Eclipse extension point. The use of the Eclipse extension point registry provides the ability for objects to dynamically be added and removed from the JNDI registry by providing extension points as a part of the `plugin.xml` files of installation artifacts.

This JNDI binding extension point is called `com.ibm.pvc.jndi.provider.java.binding`.

An example of the usage of this extension point would be similar to the following:

```
com.ibm.pvc.jndi.provider.java.binding">
    <binding
    jndi-name="java:comp/env/jdbc/dsname"
 objectFactory-id="com.ibm.pvc.jndi.provider.java.genericobjectfactory">
    </binding>
  </extension>
```

Note that a required component of the `com.ibm.pvc.jndi.provider.java.binding` extension point is the `objectFactory-id`. The WED client provides two `ObjectFactory` implementations:

- `EJBObjectFactory`
- `GenericObjectFactory`

The `EJBObjectFactory` is used exclusively for embedded transaction container bundles, while the `GenericObjectFactory` allows for an xml description of any Java object, including primitive constructor parameters, and the ability to call methods on the object once it has been created, but before it is bound into the JNDI registry and returned to a client application.

Note: No matter what specific object factory is used, all JNDI objects declaratively described are required to provide the `com.ibm.pvc.jndi.provider.java.binding` in their `plugin.xml` files. Some object factories may also provide another extension point to that needs to be implemented as well, such as the `GenericObjectFactory`.

Packaging

The schema files needed for development are included in the `com.ibm.pvc.wct.extension.schemas` plug-in which is part of the WEC Toolkit installation package and therefore is available for use in the Rational development environment.

The `ObjectFactory` implementations are delivered via two different plug-ins. The `GenericObjectFactory` is shipped as a part of the JNDI provider (`com.ibm.pvc.jndi.provider.java`) plug-in, while the `EJBObjectFactory` is packaged with the Embedded Transaction Container runtime (`com.ibm.pvc.txncontainer`) - because it is only useful when dealing with the embedded transaction container.

The availability of these object factory implementations in terms of platform deployment is inconsequential, as they are delivered with the base components as needed. If other object factory implementations are developed, care will need to be taken to ensure that they are available on the client runtime as needed, as the `plugin.xml` references alone will not cause any dependencies to be registered during deployment which could result in JNDI definitions on the runtime platform with no associated object factory to bind the object into JNDI.

Updating application features

This section describes how to update an application on the user's machine. Updating an application is similar to the initial installation. You can update an application locally, or from a Web update site.

WebSphere Everyplace Deployment uses the `eclipse` and`rcp/eclipse` directories within the WebSphere Everyplace Deployment root for its own features and plug-ins. The `shared` directory is provided as an Eclipse extension, and the default installation location for new features. This is also true for installation configurations that specify `serial.multiuser`. For installation configurations that specify `user`, the default location is in the `<user_home>`.

When new versions of features are provided, they are installed into the same directory as the previous version. The installation directory for a feature update cannot be changed. Some plug-ins have a **colocation-affinity** attribute specified. In this case, the feature and its plug-ins are installed to the location specified by the **colocation-affinity** attribute.

Versions for features are specified using **major.minor.service** qualifier. For example, a version of 4.0.1 has a major version of 4, a minor version of 0, and a service version of 1. An equivalent version is a version that differs first at the service level. A compatible version is a version that differs first at the minor level. For example, using our version 4.0.1 in the previous example, a version of 4.0.2 would be an equivalent version, because the service value is the first value that changed. A version of 4.1.2 would be a compatible version, because the minor value is the first value that changed.

The Scan for Updates capability provided as part of the Application Manager dialog enables updates of only equivalent or compatible versions, according to the preferences selected in the Install/Update dialog, which you access as follows: **Manage** → **Preferences** → **Install/Update**. The default value is for equivalent updates. Updates can only be performed here for features that have the URL and update attributes specified in their feature.xml file. For example:

```
<url>
     <update_label="Your Update Site"
     url=http://updatesite/com.your.plugin.site/site.xml>
</url>
```

Note: The URL must end in the `site.xml` file or the update will not be found. If multiple versions of a feature are included in an update site the latest version should be listed first in the `site.xml` file. Because of a bug in eclipse only one of the available versions will show as available.

In addition, features that are a part of the core product will be filtered out in the Application Management dialog and the user will not be able to update using the Application Manager dialog. A user can update the core product using the Install dialog. This requires that the user is provided a site URL to enter on the first screen.

A feature version that changes at the major level, for example, a version 5.0.0 that would replace our version 4.0.1 in the previous example, must be installed through the **Application** → **Install** mechanism. Scan for Updates will not show this feature as being available.

For additional information on versioning, refer to the Feature manifest section of the Platform Plug-in Developer's Guide.

Troubleshooting installation problems

- Q: **During test installation of the application feature(s), I get an error message indicating that either <*Feature*> requires <*plug-in id*> plugin or <Feature> requires <*feature*> feature, and my installation is not allowed to continue. How do I correct this situation?**

  Features have options that allow specification of both other included features, or features that are expected to be available on the platform, or plug-ins that are expected to be available on the platform. If you see this message, it indicates that a feature or plug-in that is expected to be on the platform is not currently installed.

  If there are other features listed on your update site, you might be able to select one of the other features, allowing the required features to resolve. If there are no other features, or selecting other features on the update site does not eliminate the message, then you will need to locate the required feature to install the application.

  Check to see if the required feature is one that was provided on the WebSphere Everyplace Deployment distribution media. If you can locate it there, you will need to include that feature on the update site that you are using for installation. If you cannot locate it there, check with the application provider to see if they can provide a missing feature.

  If you have created your own update site, check to see that you have included all features for the application on the update site.
- Q:**During test installation of the application feature(s), I get an error message indicating Unable to retrieve remote reference <*location*> and my installation is not allowed to continue. How do I correct this situation?**

  Features have options that allow specification of other included features, or features that are expected to be available on the platform. If you see this message, it indicates that a feature that is included in your main application feature is not currently available.

  Check to see if the referenced feature is one that was provided on the WebSphere Everyplace Deployment distribution media. If you can locate it there, you will need to include that feature on the update site that you are using for installation.

  If you have created your own update site, check to see that you have included all features for the application on the update site.

  If you received a prebuilt update site, check to see if there is a prerequisite that some features are already installed on the platform.
- Q:**During test installation, the feature installed, but my application either does not show up in the application list, or fails when I attempt to use it. What is wrong?**

  The application that you installed contains a set of plug-ins incorporated into the features. There may be cases where the plug-ins have specific runtime dependencies that are not met. Preferably, runtime dependencies for plug-ins would have been expressed within the feature, but that is not always the case. Through the use of the WebSphere Everyplace Deployment console, the Platform Manager tool, or the Admin Utility for OSGi you can view the status of the plug-ins in the platform. (The Platform Manager tool and the Admin Utility for OSGi are separately installable applications available on the predefined WED Evaluation Components update site). The Platform Manager allows you to display status organized by feature or by application. Use this view to see the associated plug-ins. If the plug-ins show a status of INSTALLED but not RESOLVED or ACTIVE, then there are missing dependencies. Select any installed bundles, then select the Diagnostics button to show any unresolved constraints. You will need to correct the problems identified here before your application is usable. Once you have identified the resolved the problems, if you have made changes to your application, you will need to uninstall the existing version and install a new version of the application.

- Q: **When installing as a system management service, install states that a service already exists. What can I do?**

  Only one instance of the WebSphere Everyplace Deployment service can exist on a system. Uninstall WebSphere Everyplace Deployment and try again. If WebSphere Everyplace Deployment has been uninstalled or accidentally deleted (so that the uninstall option is not available), then remove the service using the following instructions:

  On Windows:

  1. Open a command prompt
  2. Run `sc sdshow WEDMgmtService`.
  3. If the service exists, run `sc delete WEDMgmtService` to remove the service`

  On Linux:

  1. Run ``killall `pidof mgmtservice` ``
  2. Delete the `/etc/init.d/mgmtservice` file

Managing applications with a Device Management Server

There are three ways that you can deploy applications to be installed on the user's machine. You can deploy locally, from a Web site, or through a Device Management Server. This section describes how to use the Enterprise Management Agent and Device Management Server to deploy applications.

Refer to "Managing applications with WebSphere Everyplace Deployment" on page 55 for specific instructions on how to prepare an application for deployment using either local or Web deployment.

System administrators have additional capabilities to manage the platform from a central server environment:

- Install features (applications) into user runtimes
- Install native (Windows or Linux) applications
- Collect hardware and software inventories
- Collect, review, and manage configuration values

For workstations using the WebSphere Everyplace Deployment standard installation running a workbench, installing features (applications) is one of the primary tasks. Installing native Windows and Linux applications is a secondary task. For workstations using WebSphere Everyplace Deployment as a system management service, installing native Windows and Linux applications is the primary task. Installing features into the service runtime is a secondary task.

Regardless of the type of application that you want to install, the set of steps required is very similar. Both application installation tasks make use of the same NativeAppBundle command line utility, but with different parameter sets.

Refer to "Installing native applications using a Device Management Server" for instructions on installing native applications.

Other tasks such as hardware and software inventories, and configuration management are covered in "Managing client runtimes with a Device Management server" on page 41.

Installing native applications using a Device Management Server

The NativeAppBundle tool is provided as part of Device Management Server distribution. Because a Device Management Server distributes only bundles when using the Enterprise Management Agent, the NativeAppBundle tool provides the capability to provide a bundle wrapper for a native application that is to be installed on the target device.

Refer to your Device Management Server documentation for more information on using the NativeAppBundle tool and distributing native applications using a Device Management Server.

Note: "Allow Service to Interact with Desktop" must be selected in the Service Control Panel for WED Management Service" for Native Application GUI installs to work.

Note: Enabling an interactive desktop may introduce security holes. This makes it possible for another application running on the desktop to interact with the service. Also note that ″Allow Service to Interact with Desktop″ will not work with fast-user switching, Remote Desktop, or Terminal Services. This option can only be utilized by an administrator logging in locally to the target machine.″

Refer to "Installing for system management" on page 12 for more information.

Preparing Eclipse features for deployment

The NativeAppBundle tool is provided as part of the Device Management Server distribution. Because the Device Management Server distributes only bundles when using the Enterprise Management Agent, the NativeAppBundle tool provides the capability to provide a bundle wrapper for a set of files that are to be installed on the target device.

To distribute software to the client platform, it is recommended that you use the NativeAppBundle tool to wrap an Eclipse update site. When the created bundle is received on the client by the agent and installed, it invokes the Update Manager to install the features defined in the update site. Because the distributed bundle contains target file names, you will need to create separate distribution bundles for Windows and Linux targets.

The following steps are required to create and distribute an update site:

1. Create an update site in a directory on your system (a developer or software vendor should provide this for you).
2. Run the NativeAppBundle tool to create the bundle.
3. Register the bundle with the Device Management Server.
4. Create software distribution jobs to distribute the bundle.

The section "Installing the Order Entry Sample using a Device Management server" on page 65 provides step-by-step instructions to illustrate these steps.

Note: Each bundle that you define using the NativeAppBundle tool should use a unique bundle name. For example, if you have created a distribution bundle with a `BundleName` of `MyApp`, containing version 1.0.0 of your application, you should use a different `BundleName` (for example, `MyApp_v2`) when creating a distribution bundle containing version 2.0.0. This refers only to the `BundleName` specified as parameters to the `NativeAppBundle` program, not to the features or plug-ins contained in the update site.

The following information contains specific and recommended settings for using the NativeAppBundle tool to distribute software to the WebSphere Everyplace Deployment client platform. Refer to the WEDM documentation section on Native Software Distribution Jobs with OSGi bundles more information on using the NativeAppBundle tool and distributing native applications using the Device Management Server.

```
NativeAppBundle -BundleName=bundle_name
                -InputDirectory=update_site_directory
                -InstallDirectory=temporary_ install_directory
                -BuildDirectory=target_output_directory
                -Eclipse=default
```

```
-JarName=jarfilename
-BundleVersion=bundleVersion
-CleanupAfterInstall=yes
-RemoveOnUninstall=no
```

Notes:

1. `bundle_name` is a name for the bundle that you define. The bundle name may not contain spaces.
2. `update_site_directory` is the update site directory.
3. `temporary_install_directory` is the name of a directory that will be used on the client, and is operating system specific. This value may contain a variable reference that will be replaced on the client during bundle installation. The manifest file generated for this bundle uses `%` as the delimiter surrounding the variable name. The search order for the variable value is the operating system environment, followed by Java System properties.

   Note that if you use the Windows command line to specify a variable name (with `%` delimiters), you will need to add escape characters. For example, to use the value of the environment variable `MY_TEMP_DIR` on the client system, you would need to specify `-InstallDirectory=^%^%MY_TEMP_DIR^%^%`. If you use the `-Parameters` option to specify a text file containing the options, you do not need to escape the `%` character (you would specify only `-InstallDirectory=%MY_TEMP_DIR%`).
4. The `-eclipse` parameter defines the Eclipse extension directory into which the features will be installed. The special value `default` will install the features into the current location for WebSphere Everyplace Deployment.
5. `target_output_directory` is the location where the created JAR should be written.
6. `jarfilename` is a file name to be used for the Jar (optional, default is the bundle_name will be used for the JAR.
7. `bundleVersion` is the version for the JAR (optional, default is 1.0.0).

If `-Eclipse` is used, the target bundle is targeted specifically for installation into an Eclipse extension framework.

Installing the Order Entry Sample using a Device Management server

The steps in the sections below provide an example of distributing software to the client platform using the Device Management server. Prior to beginning the steps, you will need to ensure that you have satisfied the following prerequisites:

- You have installed a Device Management server using one of the available product installations.
- You have an HTTP (or FTP) Server available on which you can put and get files.
- You can start the Device Manager Console program.
- You have a client platform installed on a system that you are able to use for testing.
- You have the installation media that provides the client platform. The distribution task will reuse the JARs provided on the update site.

The following tasks guide you through the distribution of the Order Entry Sample application using the Device Management Server:

1. “Creating Order Entry for distribution using the NativeAppBundle tool” on page 67
2. “Creating the Software Distribution job for the Order Entry feature” on page 68
3. “Running the Order Entry sample applications” on page 69

Note: There are two users that are identified in the subsequent instructions. The first user identity is referred to as **client01** with a password **client01**. This is the identity that the Enterprise Management Agent will use to connect to the Device Management Server.

The second user identity the identity that an administrator would use to access the DM Console. In a WebSphere Everyplace Deployment server, use the WebSphere Everyplace Deployment user ID and password. In a WEDM server, the default subscription settings for the user ID and password are **dmadmin** with a password of **dmadmin**.

In both cases, replace the user ID and password with a user ID and password appropriate to your environment.

Enrolling the client with a Device Management Server

This section discusses enrolling your WebSphere Everyplace Deployment client with the Device Management Server.

1. Start WebSphere Everyplace Deployment.
2. Select **Manage > Preferences > Enterprise Management Agent**.
3. Select the **Enable Enterprise Management Agent** option.
4. A dialog box will pop up asking you to confirm that you want to enable the Enterprise Management Agent. Select **OK**.
5. Enter the following parameter settings:
   a. Device User Name: **client01**
   b. Device User Password: **client01**
   c. Re-enter Device User Password: **client01**
   d. Server IP Address: **<device management server hostname>**
   e. Polling Configuration:

      The polling configuration is provided here for information only. It is not changeable here.

      The Enterprise Management Agent contacts the Device Management Server based on a start and stop polling window. When the Enterprise Management Agent is enabled for the first time it will attempt to contact the Device Management Server regardless of the polling window settings. Each subsequent restart, the agent will not contact the Device Management Server if it is outside the polling window.
6. Click **OK**.

The platform will begin the process of connecting to the Device Manager and enrolling the agent/platform with the server.

Once the agent has successfully enrolled with the server, a message is added to the log file:

```
Connection to the DMS server was successful. Please contact the DMS administrator
if you wish to disable Enterprise Management Agent.
```

Creating Order Entry for distribution using the NativeAppBundle tool

Before you can distribute applications using a Device Management Server, you must convert it to a software bundle format using the NativeAppBundle tool. The NativeAppBundle tool takes an Eclipse update site as its source and creates a bundle containing all features on the update site. You will need to use the WebSphere Everyplace Client Toolkit within a Rational Software Development Platform, such as Rational Application Developer, to create a custom site containing the Order Entry sample.

Follow these steps to create a custom update site using a Rational Software Development Platform:

1. Launch your Rational Software Development Platform.
2. Install the WebSphere Everyplace Client Toolkit if it is not already installed.
3. Select Help.
4. Select Samples Gallery.
5. Select Showcase Samples.
6. Select WebSphere Everyplace Deployment, and then select Order Entry Applications.
7. Follow the directions on this page to set up and import the Order Entry Applications into your workspace. Several projects are created in your workspace.
8. Ensure that there are no compile errors, then create a Feature project by clicking **File → New → Feature Project**. Your feature should contain the following four plug-ins:
   - com.ibm.pvc.samples.orderentry.common
   - com.ibm.pvc.samples.orderentry.richapp
   - com.ibm.pvc.samples.oderentry.service
   - com.ibm.pvc.samples.orderentry.webapp
   -
9. Create an update site project by clicking **file → New → Other → Plug-in Development → Update Site Project**. The update site project should contain the feature you created in the previous step. This is the update site that you will use as input to the NativeAppBundle tool.

Now that you have created a custom update site, follow these steps in the DM Console on the Device Management Server system to create the software bundle with the NativeAppBundle tool:

1. Change to the `<DM Server install directory>/bin` directory.
2. Run the command to create your software bundle.

```
NativeAppBundle -BundleName=Order_Entry_Feature
                -InputDirectory=<location of your custom update site>
                -InstallDirectory=c:/Program Files/IBM/WED/temp
                -BuildDirectory=c:/Program Files/IBM HTTP Server/htdocs
                     /en_US/bundles
                -Eclipse=default
                -JarName=OrderEntryFeature
                -BundleVersion=6.0.0
                -CleanupAfterInstall=yes
                -RemoveOnUninstall=no
```

Notes:

a. <*location of your update site*> is the custom update site you have created, which contains only the Order Entry sample.

b. `c:/Program Files/ IBM HTTP Server/htdocs/en_US/bundles` is assumed to be a directory that is served by an HTTP server and from which the file can later be accessed. Note: the bundles directory is not normally there. You should have created it when you set up your HTTP Server document root.

c. `c:/Program Files/IBM/WED` is assumed to be the install directory for WebSphere Everyplace Deployment.

Registering Order Entry with a Device Management server

Once the distributable form of the Order Entry application has been created with the NativeAppBundle tool, it needs to be registered (defined) to the Device Management server before it can be distributed.

1. Start the Device Manager(DM) console.
2. Login into the DM console using the following set of parameters:
   a. User ID: **dmadmin**
      - In a WebSphere Everyplace Deployment server, use the WebSphere Everyplace Deployment administrator user ID.
      - In a WEDM Server, the default subscription manager setting is **dmadmin.**

   b. Password: **dmadmin**
      - In a WebSphere Everyplace Deployment server, use the WebSphere Everyplace Deployment administrator password.
      - In a WEDM Server, the default subscription manager setting is **dmadmin.**

   c. Device Manager Server: **<DM server hostname>**
3. Add the OrderEntryFeature bundle to the Device Management Server software inventory:
   a. Right click on Software and select **New Software**.

   b. Select the software type for your client environment. Possible values are:
      - NativeOSGiBundle
      - NativeWin32OSGiBundle
      - NativeLinux_x86OSGiBundle

      .

   c. Specify the location (`http://` or `ftp:/ URL`) of the `OrderEntryFeature+6.0.0.jar`.

      For example, assuming that the HTTP Server document root on a Windows installation is set to `c:/Program Files/IBM HTTP Server/htdocs/en_US`, then the URL would be `http://<servername>/bundles/OrderEntryFeature+6.0.0.jar`.

   d. Select **Fetch**.

   e. Select **Next**.

   f. Select all of the operations for the `Device Class = OSGi`, then click **OK**.

   g. The bundle will be added to the DM server software inventory.

Creating the Software Distribution job for the Order Entry feature

Follow these steps in the DM Console on the Device Management Server system to create the Software Distribution job for the Order Entry feature.

1. Select **Devices**.
2. Select **Use New Query** and **Return anything** as your search criteria.
3. Select **OK**.
4. The DM console will show a list of enrolled devices.
5. Right click your device, and select **Submit Job**.
6. Select **Next**.
7. Select the **Job Type** as Software Distribution (use default settings for all the other job attributes).
8. Select **Next**.
9. Select **Add Group**.
10. Select **Order_Entry_Feature** from the list of Available Software.
11. Set Auto start to **true**.
12. Select **Next**.
13. Select **OK**.
14. The job has been submitted. Select **Close**.
15. Check the status of the Job at the DM Console.

Running the Order Entry sample applications

Once the distribution job has completed, the features provided in the update site have been installed onto the file system on the client. Run the Order Entry sample applications by performing the following procedure on the client side:

1. Stop and restart WebSphere Everyplace Deployment. The installed features will not appear until you do so.
2. Once you have stopped and restarted, the Order Entry Web Application and Order Entry Rich Client Application will be added to **Application > Open** menu.
3. Launch one of the two applications.

For more information on using samples, visit the Samples Gallery in Rational Application Developer.

Removing the Order Entry Sample

To remove the Order Entry sample, perform the following procedure within the DM Console on the Device Management Server system:

1. Select the **Device**.
2. Right click, then select **Submit Job...**
3. Select **Next**.
4. Select **Bundle Control** as the Job Type.
5. Select **Next**.
6. Select **Add Step**.
7. Select **Uninstall** for the Job Type.
8. Select **Order_Entry_Feature** from either the ’OSGi Bundles from Inventory field’ or the ’OSGi bundles from repository not listed in inventory’ selection list.
9. Select **Next**.
10. Select **OK**.
11. Select **Close**.

The feature will be installed when the job is received at the client. Stop and restart the client after the feature has been uninstalled.

Configuring security

WebSphere Everyplace Deployment includes the IBM J2SE 1.4.2 JVM on the Windows and Linux platforms. The JVM includes a set of default JSSE and JCE providers that are available for application usage.

Refer to the following URL for information about the security capabilities of the JVM on Windows:

`http://www.ibm.com/developerworks/java/jdk/security/142/secguides/securityguide.win32.html`

Refer to the following URL for information about the security capabilities of the JVM on Linux:

`http://www.ibm.com/developerworks/java/jdk/security/142/secguides/securityguide.lnx.html`

http://www.ibm.com/developerworks/java/jdk/security/142/ is the index page for information on the security aspects of the IBM J2SE 1.4.2 JVM.

The use of JCE within the platform follows the standard patterns of JCE usage.

Because the platform uses multiple inbound and outbound communication mechanisms, depending upon the type of communication required, there are specific configuration steps needed to configure JSSE to support SSL communications. Refer to the appropriate section according to the desired communication mechanism.

WebSphere Everyplace Deployment does not support the use of Java 2 Security to grant or prevent code access based upon identity.

Configuring SSL for the platform

To configure the default configuration, you must start the VM with the appropriate system properties to provide a valid client certificate. Refer to "Configuring Java system properties" on page 24 for information on how to set the properties.

You can specify the following properties for the client:

```
-Djavax.net.ssl.keyStore=<path_to_keystore_file>
-Djavax.net.ssl.keyStoreType=<keystore_type>
-Djavax.net.ssl.keyStorePassword=<keystore_password>
-Djavax.net.ssl.trustStore=<path_to_truststore_file>
-Djavax.net.ssl.trustStoreType=<truststore_type>
-Djavax.net.ssl.trustStorePassword=<truststore_password>
```

For more information on setting these properties, and configuring SSL for the JVM, refer to the following URL:

`http://www.ibm.com/developerworks/java/jdk/security/142/secguides/jssedocs/JSSERefGuide.html`

Configuring SSL for the Enterprise Management Agent

You can configure the Enterprise Management Agent to use SSL to communicate with the Device Management Server.

The Enterprise Management Agent expects that a truststore (JKS format) file with a certificate is provided for SSL communication. For instructions on creating a file that matches the server certificates refer to your Device Management Server documentation.

Note: When you create the certificate, the Common Name (″cn=″ value) field you specify must match the server address the Enterprise Management Agent uses to connect with the Device Management Server.

The Enterprise Management Agent will look in three locations for the truststore file and takes the first available location. The Enterprise Management Agent searches the three truststore locations in the following order:

1. A file with the name and path specified in the Config Admin (CM), which you can fully customize with a DMS job.
2. A file named `key.jks` in the path specified by the System Property **user.home**.
3. A file named `key.jks` in the path specified by the directory you are working from, ″./″.

If the Enterprise Management Agent finds a file in locations #2 or #3, it will set the CM location property to that location. The Enterprise Management Agent assumes the default truststore password is ″password″. This password value is also stored in CM and can be modified using a DMS custom command job.

You can manage the SSL Configuration with the DMS Node Discovery and Custom Command jobs. The base Note to discover from is ./OSGi with a depth of 4.

The SSL Enabler bundle manageable nodes are:

- `com.ibm.pvc.osgiagent.protocol.agentadaptor.https` - the OSGi SERVICE.PID that identifies the HTTPS Adaptor implementation.
- `SSLEnabler.trustloc` - the truststore file path
- `SSLEnabler.trustpassword` - the truststore file password
- `SSLEnabler.mode` - the security mode. Possible values are ″trustall″ or ″certificates″

By setting the SSL Mode to `trustall`, the client ignores certificates and trusts all HTTPS connections. This does not necessarily mean that the server will trust the client. You can only configure the server to trust the client through the Device Management Server.

Configuring SSL for Web Services

If you plan on running Web Services applications that connect to Web Services located behind a secure URL, for example HTTPS, you must set up WebSphere Everyplace Deployment with an appropriate default configuration. The Web Services runtime does not provide any SSL specific configuration capabilities and relies on the default platform settings. Refer to “Configuring SSL for the platform” on page 71 for more information.

Configuring SSL for the Web Container

To enable the web container to serve requests using a secured socket, perform the following steps:

1. Configure the web container.
   a. Create a SSL configuration file and set the following properties

Table 6. SSL configuration properties

Property	**value**	**optional/mandatory**	**Description**
com.ibm.ssl.contextProvider	IBMJSSE2 [default], IBMJSSE, IBMJSSEFIPS	optional	Specifies the SSL provider
com.ibm.ssl.protocol	SSL [default], SSLv2, SSLv3, TLS, TLSv1	optional	Specifies the protocol to use
com.ibm.ssl.keyStoreType	JKS [default], JCEK, PKCS12	optional	Specifies the type of keyStore to use
com.ibm.ssl.keyStore	Fully-qualified path to file on the filesystem	mandatory	Specifies the location of keyStore on the file system
com.ibm.ssl.trustStoreType	JKS [default], JCEK, PKCS12	optional	Specifies the Password to open the keyStore
com.ibm.ssl.trustStore	Fully-qualified path to file on the filesystem	mandatory	Specifies the type of trustStore
com.ibm.ssl.keyStorePassword	Plain-text password for keyStore	mandatory	Specifies the location of trustStore on the file system
com.ibm.ssl.trustStorePassword	Plain-text password for trustStore	mandatory	Specifies the password to open trustStore
com.ibm.ssl.clientAuthentication	True | false	Mandatory	Specifies whether or not the client application requires authentication. The default is false.

   b. Set the system property`com.ibm.pvc.webcontainer.ssl.configfile` to point to the SSL configuration file created in (a)
   c. Set the HTTPS port using ConfigurationAdmin or using either of the system properties `com.ibm.pvc.webcontainer.port.secure` and `com.ibm.osg.webcontainer.port.secure`

      Note: The web container will only secure requests if the HTTPS port is set and the ssl configuration file is supplied to the web container. If the port is not set, the web container will default to running the requests on the HTTP port.
2. Create the external keyStore and trustStore repositories:
   a. Use iKeyman or the keytool command-line utility (comes with the JDK) to create the keyStore and trustStore files. Location of these files is specified using the SSL configuration file.
   b. Use iKeyman or the keytool command-line utility to generate key entries and certificates and store them in the keyStore and trustStore files.

When the web container is configured to enable SSL support, the web container will delete the SSL configuration file and store the configuration in a web container.properties file (passwords are encrypted) in the user’s configuration directory.

Configuring SSL for ISync

The DB2® Everyplace Sync technology (ISync) allows for the use of SSL to connect from the client to the DB2 Everyplace Sync Server, providing that both client and server are SSL-enabled in a compatible configuration. When using the pure Java IBM Cloudscape ISync implementation, there are no ISync specific SSL configuration steps required. The IBM Cloudscape Java ISync provider assumes that the client platform has already been configured to support the SSL outbound connections. Refer to "Configuring SSL for the platform" on page 71 for information on setting up SSL.

When using the native DB2 Everyplace native database and synchronization implementation, some operating system specific tasks are required in order to prepare the client for synchronization with the DB2 Everyplace sync server. Refer to the Configuring SSL for DB2 Everyplace client devices section of the IBM DB2 Everyplace Installation and User's Guide version 8.2, which is included in the Developing data access and synchronization applications chapter of the WebSphere Everyplace Deployment Developer's Guide.

Using SSL from applications

Creating SSL connections to servers

The default platform configuration for SSL for creating connections to servers is handled as described in "Configuring SSL for the platform" on page 71. Applications that need to create their own SSL connections to a server can make use of the same platform configuration.

Applications need only to create a new URL specifying HTTPS as the desired protocol, and the appropriate HttpsUrlConnection object will be created. Applications should either rely on the default configuration, or configure the security information on a per instance basis. Changing the default configuration may have adverse effects on other applications running in the platform, and on the connection capabilities of the components provided with the platform.

Creating SSL sockets for incoming connections

The default SSLServerSocketFactory that provides SSLServerSockets is based upon the settings provided in the java.security file for the platform. Refer to the articles listed in "Configuring security" on page 71 for more information on when and how to update this file. Applications should rely on the default configuration.

Applications attempting to open SSLServerSockets for their own usage should be aware that the web container component of the platform may also attempt to open SSL Server Sockets. Changing the default configuration may have adverse effects on other components provided with the platform.

Enabling FIPS compliant JCE and JSSE providers

This section describes how to enable Federal Information Processing Standards (FIPS) compliant JCE and JSSE providers.

Enabling use of FIPS compliant JCE and JSSE providers requires updates to the `java.security` configuration file.

The `java.security` configuration file is contained in the following directory on Windows:

```
<installation directory>\rcp\eclipse\plugins\com.ibm.rcp.j2se.win32.x86_1.4.2.SR2\jre\lib\security
```

The`java.security` configuration file is contained in the following directory on Linux:

```
<installation directory>\rcp\eclipse\plugins\com.ibm.rcp.j2se.linux.x86_1.4.2.SR2\jre\lib\security
```

Refer to http://www.ibm.com/developerworks/java/jdk/security/142/FIPShowto.html for more information on enabling FIPS compliant providers.

Problem determination

Logging

You use logs to gather information about problems that might happen while using WebSphere Everyplace Deployment for Windows and Linux. This section describes how to access logs, adjust logging levels, and configure how WebSphere Everyplace Deployment manages log files.

Logging framework

The OSGi framework, Eclipse framework and the 1.4 JDK (JSR47) all provide different systems for logging messages. Applications in some cases also leverage standard error and standard out for message delivery, and these messages must also be captured to ensure all data is available during the problem determination stage. The WebSphere Everyplace Deployment runtime `com.ibm.pvc.wct.internal.logredirector` plug-in, hereafter referred to as logRedirector, provides the ability to collect all messages logged in the WebSphere Everyplace Deployment runtime into one persistent log file. The logRedirector captures messages logged from the OSGi logService, the Eclipse logging APIs, and standard error and standard out and redirects them to the JDK 1.4 `java.util.logging`. A java.utils.logging log file manager is also provided with the WebSphere Everyplace Deployment runtime (`<`*`installation directory`*`>/rcp/loggerboot.jar`) which supports configuration of the JDK logging and manages the persistent log file in `<USER_HOME>\IBM\RCP\<INSTALL_ID>\<USER_NAME>\logs\rcp.log.*` where * represents the different generations of log files, such as rcp.log.0.

Each of the logging systems available in the WED runtime has its own definition of logging levels. In order to bring all of the messages from these disparate systems together a mapping was needed between the logging levels. The following section describes this mapping.

OSGi to JDK level mapping:

Because `java.util.Level` doesn't provide a DEBUG level, OSGi DEBUG messages will be written as FINEST messages. ERROR messages are mapped to SEVERE messages. INFO and WARNING messages are mapped directly.

Table 7. Log level mapping between the OSGi log and the java.util.Level

OSGi Log Level	java.util.Level
ERROR	SEVERE
WARNING	WARNING
INFO	INFO
DEBUG	FINEST

Eclipse to JDK level mapping:

Table 8. Log level mapping between the Eclipse log and the java.util.Level

Eclipse Log Level	java.util.Level
CANCEL	SEVERE

Table 8. Log level mapping between the Eclipse log and the java.util.Level (continued)

Eclipse Log Level	java.util.Level
ERROR	WARNING
WARNING	INFO
OK	FINEST

Manually adjusting the logging level

The logging framework default `java.util.logging` configuration properties are stored in the `plugin_customization.ini` file, which resides in the `<`*`installation directory`*`>/rcp directory`. The properties are in <key, value> format, and are set to the following by default:

Table 9. Default log properties at startup time

Properties	plugin_customization.ini default values
Handlers	java.util.logging.ConsileHandler com.ibm.rcp.core.logger.boot.RCPFileHandler
.level	WARNING
com.ibm.rcp.core.logger.boot.RCPFileHandler.level	FINEST
com.ibm.pvc.wct.internal.logredirector.level	FINEST

The `plugin_customization.ini` file allows one level of customization of the WebSphere Everyplace Deployment runtime logging framework. All properties defined for `java.util.logging.LogManager` are supported. This file can be used to set additional or alternate handlers, log levels for handlers and for loggers. Default handlers are: `java.util.logging.ConsoleHandler` and `com.ibm.rcp.core.logging.RCPFileHandler`. To replace these handlers use a property such as:

```
handlers=<a list of handlers separated by white spaces>
```

To add additional handlers to the default simply include the default handlers in the list.

Logger namespaces are hierarchical so `com.ibm.level=SEVERE` will turn off all of our logging below SEVERE by default and then you can override this for individual points in the hierarchy that you are interested in. The following is an example of typical levels:

```
com.level=INFO
com.ibm.level=FINEST
```

The first entry sets the level for the logger for the com package and all loggers created from packages that are children of it to INFO. The second entry overrides this setting for the logger created for the package com.ibm, setting it to FINEST.

For more information on configuring the JDK logger please refer to the JDK documentation for the java.util.logging package.

The **logRedirector** component of the WebSphere Everyplace Deployment runtime also allows for configuration of the level of Eclipse Log and OSGi Log Service messages that are sent to the console and to the persistent log file. The system property `-Dlogredirector.level` is read to configure the logging level of the logRedirector. This can be added into the <*installation directory*>/rcp/rcpinstall.properties file in order to have the property set at platform launch time. If this property cannot be accessed, the logRedirector will use the default setting "WARNING".

The `logredirector.level` defines the severity level of log entries that will be redirected to the WED log. All entries with a severity greater than or equal to the value will be written to the log. The values accepted are based on the java.util.logging defined constants provided here in increasing severity order: FINEST, FINER, FINE, INFO, WARNING, SEVERE.

The standard error and standard out messages are also logged to the persistent log file and sent to the console. To configure what level the standard error and standard out messages should be logged to the JDK logger, the following two properties can be set: `logredirector.err.level` and `logredirector.out.level`. These can be set in the `rcpinstall.properties` listed above, and both default to INFO.

Log file management

This section provides information on how WebSphere Everyplace Deployment manages the log file.

The persistent log file for the WED platform is managed by a `java.util.logging FileHandler`. The `logger.properties` file located in the <*installation directory*>/rcp/ directory can be used to change the management policy for the log file. The default settings for the log file manage are shown in the following table.

Table 10. Log properties and their default values

Configuration variables	**logger.properties default values**
log.append	false
log.generations	12
log.size	2000000
logfile.formatter	java.util.logging.SimpleFormatter

If `log.append` is set true, the new messages will be written from the end of the log file, while the default is to create a new log file each time the runtime is started. The maximum number of generations of log files is 12. Any setting that is bigger than 12 is treated as 12. The maximum size of log file is 2000000. Any setting that is bigger than 2000000 is treated as 2000000.

Note: A value of 0 for the size will be treated as an unlimited file size. If you would like to stop all logging from the WebSphere Everyplace Deployment runtime to the file system, you can change the `com.ibm.rcp.core.logger.boot.RCPFileHandler.level` entry of the <*installation directory*>/rcp/plugin_customization.ini file from its default of FINEST to OFF. When you have updated this setting, and restarted the platform, logging data will no longer be written to a persistent file.

The `logfile.formatter` sets the log file format, which defaults to the `SimpleFormatter`. For more information on these configuration options, please refer to the `java.util.logging.FileHandler` JDK documentation.

Tracing

Several of the WebSphere Everyplace Deployment runtime components, such as the Embedded Transaction Container, and the Web Container, have detailed tracing capabilities for enhanced problem determination in specific areas of the runtime. For specific trace enablement information for the WebSphere Everyplace Deployment runtime components please refer to the specific documentation for each component.

The Enterprise Management Agent provides extensive tracing for debugging purposes. For WebSphere Everyplace Deployment 6.0 a Java system property is used by the agent to enable the debug tracing.

Add the following system property to enable debug tracing.

```
com.ibm.osg.service.osgiagent.osgiagent.debug=true
```

For the changes to take effect, the runtime client must be restarted.

Refer to "Configuring Java system properties" on page 24 for more information.

Using the IBM Support Assistant

IBM Support Assistant is designed to help answer questions and gather data for service personnel if needed.

IBM Support Assistant provides the following functions:

- The ability to perform a federated search across information repositories
- Convenient access to support-related web resources
- A way to automate data collection and transmission to IBM to expedite problem resolution.

The search component can be used to execute a federated search that concurrently accesses multiple search locations and returns results in a hierarchical arrangement. The support links component is a consolidated list of IBM web links organized by brand and product. The service component allows for the gathering of problem determination data and assists in the process of creating a problem management record with IBM Support.

You access the IBM Support Assistant from WebSphere Everyplace Deployment by selecting the Help menu option, then selecting IBM Support Assistant. A User's Guide is available from the Help tab on the navigation bar of the IBM Support Assistant page.

To capture problem determination:

1. When you have opened the IBM Support Assistant select the **Service** tab.
2. Click **Invoke Collector**. ISA gathers the problem determination information. When the collector completes a panel is displayed noting the location of the archive that was created that contains all of this information. This archive will be needed for detailed problem determination with IBM Support.
3. From the Service tab, you can send the collected information to IBM Support using the Send System Data option. You select a geographic location, and the data is sent to one of the following destination servers depending upon your selection:

North America and Asia Pacific:testcase.boulder.ibm.com\ps\toibm\pvc

EMEA: ftp.emea..ibm.com\toibm\other

Data collection details:

When the collector is invoked from the Service tab of the IBM Support Assistant, the following data is gathered:

- Platform data:
  - $(InstallDir)/eclipse/configuration/*.log
  - $(InstallDir)/eclipse/configuration/org.eclipse.update/install.log
  - $(InstallDir)/eclipse/configuration/org.eclipse.update/platform.xml
  - $(InstallDir)/eclipse/configuration/org.eclipse.update/history
  - $(InstallDir)/rcp/db2j.log l $(InstallDir)/logs/*.*
  - $(InstallDir)/rcp/rcpinstall.properties l
  - $(InstallDir)/rcp/javacore*.*
  - $(InstallDir)/rcp/plugin_customization.ini
  - Java System properties
  - Features information
  - Plug-in Registry
  - User Preferences
  - Plug-in Status
  - Enterprise Management Agent Configuration Details
- User data:
  - $(Workspace)/systemsummary.txt
  - $(Workspace)/.metadata/*.log
  - $(Workspace)/.config/*.log
  - $(Workspace)/.config/org.eclipse.update/platform.xml
  - $(Workspace)/.config/org.eclipse.update/history
  - $(Workspace)/.metadata/.log
  - $(Workspace)/logs
  - $(Workspace)/data/log/*.dat
  - $(Workspace)/datastore.log
  - $(Workspace)/shutdown.log

Notices

Trademarks