Assembling and Deploying Lotus Expeditor Applications
<table>
<thead>
<tr>
<th>Migrating a WebSphere Everyplace</th>
<th>181</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment 6.0 Platform Builder project to Lotus Expeditor Client 6.1.x</td>
<td>143</td>
</tr>
<tr>
<td>Running a target runtime image</td>
<td>143</td>
</tr>
<tr>
<td>Configuring application layout</td>
<td>143</td>
</tr>
<tr>
<td>Packaging applications for deployment</td>
<td>144</td>
</tr>
<tr>
<td>Understanding feature life cycle</td>
<td>145</td>
</tr>
<tr>
<td>Creating a feature</td>
<td>147</td>
</tr>
<tr>
<td>Controlling the feature install location</td>
<td>148</td>
</tr>
<tr>
<td>Enabling features for update</td>
<td>149</td>
</tr>
<tr>
<td>Exporting a feature and its contents</td>
<td>150</td>
</tr>
<tr>
<td>Creating an update site</td>
<td>151</td>
</tr>
<tr>
<td>Using the global install handler</td>
<td>151</td>
</tr>
<tr>
<td>Creating the install handler instructions file</td>
<td>151</td>
</tr>
<tr>
<td>Creating instructions for desktop updates</td>
<td>154</td>
</tr>
<tr>
<td>Adding the global install handler to a feature</td>
<td>156</td>
</tr>
<tr>
<td>Deploying features to the platform</td>
<td>156</td>
</tr>
<tr>
<td>Using the provisioning manifest</td>
<td>160</td>
</tr>
<tr>
<td>Understanding the merge processing for the provisioning manifest</td>
<td>161</td>
</tr>
<tr>
<td>Provisioning manifest DTD</td>
<td>165</td>
</tr>
<tr>
<td>Installing additional languages content</td>
<td>170</td>
</tr>
<tr>
<td>Upgrading features on the platform</td>
<td>171</td>
</tr>
<tr>
<td>Uninstalling features on the platform</td>
<td>173</td>
</tr>
<tr>
<td>Contributing to the deferred actions list</td>
<td>173</td>
</tr>
<tr>
<td>Deploying applications through Web Start</td>
<td>173</td>
</tr>
<tr>
<td>Supported applications and environments</td>
<td>174</td>
</tr>
<tr>
<td>Differences from the Eclipse Web Start support</td>
<td>174</td>
</tr>
<tr>
<td>Multi-user support</td>
<td>174</td>
</tr>
<tr>
<td>Structure of a Client Services Web Start application</td>
<td>174</td>
</tr>
<tr>
<td>Developing a Web Start application</td>
<td>175</td>
</tr>
<tr>
<td>Exporting the Web Start application</td>
<td>175</td>
</tr>
<tr>
<td>Installing Web Start support on the client</td>
<td>176</td>
</tr>
<tr>
<td>Deploying the application on the server</td>
<td>178</td>
</tr>
<tr>
<td>Running an application through Web Start</td>
<td>178</td>
</tr>
<tr>
<td>Internationalization considerations</td>
<td>178</td>
</tr>
</tbody>
</table>

**Managing the client runtime** | 181 |
| Managing using the Device Manager server | 181 |
| Understanding managing with the Device Manager server | 181 |
| Configuring the Enterprise Management Agent | 181 |
| Using the Enterprise Management Agent servlet | 187 |
| Installing the Enterprise Management Agent servlet | 187 |
| Starting the Enterprise Management Agent Servlet | 187 |
| Modifying device information | 187 |
| Modifying Enterprise Management Agent account information | 188 |
| Modifying Enterprise Management Agent account information using the properties file | 189 |
| Connecting to the server to check for jobs | 190 |
| Monitoring Enterprise Management Agent events | 190 |
| Creating a software distribution job from the client | 190 |
| Working with Enterprise Management Agent accounts | 191 |
| Editing the Enterprise Management Agent device information | 193 |
| Changing the Enterprise Management Agent properties | 193 |
| Registering features with the Device Manager server | 194 |
| Deploying Eclipse features | 194 |
| Updating features with the Device Manager server | 194 |
| Uninstalling Eclipse features | 195 |
| Deploying native applications | 195 |
| Required and optional parameters | 195 |
| (NativeAppBundle command) | 195 |
| Job options and tasks for distributing native software bundles (OSGi bundles) | 198 |
| Parameters in a text file sample | 200 |
| (NativeAppBundle command) | 200 |
| Native software distribution jobs with OSGi bundles | 201 |
| Managing preferences | 202 |
| Querying client runtime preferences | 202 |
| Creating client runtime preferences | 202 |
| Updating client runtime preferences | 203 |
| Deleting client runtime preferences | 203 |
| Managing properties files | 203 |
| Querying property file values | 203 |
| Changing a property file value | 203 |
| Adding a new property to a property file | 203 |
| Deleting a property from a property file | 204 |
| Configuring client plug-ins with the Enterprise Management Agent | 204 |
| Configuring the Web Container using the Enterprise Management Agent | 205 |
| Managing Rich Clients using the Portal server | 205 |
| Understanding managing using the Portal server | 207 |
| Using the IBM Lotus Expeditor MyClient sample template | 208 |
| Configuring client user interface layout | 208 |
| Using the Rich Client Layout Administration Portlet | 209 |
| Administering rich client-specific page parameters | 209 |
| Creating a copy of the Rich Client View Placeholder before adding it to a page | 211 |
| Adding the Rich Client View Placeholder to a page | 211 |
| Editing common rich client properties for a portlet on a page | 212 |
| Editing or adding a feature requirement for the portlet representing an Eclipse view | 216 |
| Removing a feature requirement for the portlet representing an Eclipse view | 217 |
| Using the Managed Browser Administration portlet | 218 |
| Installing the Managed Browser Administration Portlet | 218 |
Understanding Lotus Expeditor Client runtime configuration

This section introduces the Lotus® Expeditor Client’s runtime configurations.

The Lotus Expeditor Client is built on an Eclipse platform. Eclipse is an award-winning, open source platform for the construction of powerful software development tools and rich desktop applications. Leveraging the Eclipse plug-in framework to integrate technology on the desktop saves technology providers time and money by enabling them to focus their efforts on delivering differentiation and value for their offerings. Full details on Eclipse are available at [http://www.eclipse.org](http://www.eclipse.org).

**Single user configuration**

You can install the Lotus Expeditor Client in a single or multi-user configuration.

The single-user configuration requires that you install to an area of the file system to which you can read and write. A good choice would be within your home directory space. If you choose another area outside of your home directory, other users may be able to modify your installed applications. In either case, your data and configuration information will always be stored in your home directory by default.

The configuration and workspace areas that are created in your home directory contain all the data required to customize the installation as you change it. Your configuration is defined during installation. A configuration consists of all of the applications that are initially installed and tracks any changes following the initial install. The workspace also contains plug-in metadata, log files, and application-specific data, if available.

With the Lotus Expeditor Client’s managed-only configuration policy, users’ configurations consist of only the applications and features that have been selected. Each time a new configuration is created, a provisioning cycle is started, selectively enabling every feature that is intended for this configuration. This occurs the first time that a user installs the Lotus Expeditor Client or each time a new workspace is requested.

You can create additional workspaces. This allows you to launch the same installation but each workspace can be customized for different jobs. For more information, see “Launching the Lotus Expeditor Client using a non-standard workspace location” on page 25.

A launch icon is created during the installation. The Lotus Expeditor Client installation will only create one launch icon. You can create additional launch icons that contain commands. This allows you to launch another workspace or to send information to a particular application by selecting an icon. This action can be used to open an application in another window of the running instance. Without this capability, you would have to start another instance of the Lotus Expeditor Client to launch a window.

You can also launch the Lotus Expeditor Client from the command line or a script. For more information, see “Launching the Lotus Expeditor Client on the desktop” on page 22.

Sharing a single user installation with additional users is not supported. If you need to share an installation, the multi-user installation is recommended.

**Multi-user configuration**

You can install the Lotus Expeditor Client in a single or multi-user configuration.

Multi-user configuration is intended for use in instances where multiple people share a workstation, such as a workstation used as a kiosk with individual operating system log-ins.
A user will not be able to tell the difference in the functioning of the single user configuration and the multi-user configuration. From the user's viewpoint, the operation of the Lotus Expeditor Client is the same.

In a multi-user configuration, users normally do not install applications. You, as administrator, install all applications and the users enable them into their individual configurations. The installer must be a member of the administrator group on Windows® or defined as root on Linux®.

Choose from three locations to install applications for a user: ${rcp.home}/eclipse, ${rcp.home}/rcp, and ${rcp.home}/shared. As administrator, you control all of these locations. Users must only have read and execute permissions for these locations. The task of setting directory permissions must be performed by an administrator after the installation has completed. As administrator, you can install multiple versions of the same feature. If there are multiple versions, a user is able to select which version to use.

A user can install applications to their private user site. In most cases, it is recommended that you, as administrator, install these applications to the shared site.

As administrator, you should only install applications when there are no users using the Lotus Expeditor Client.

Follow these administrative steps for configuring for multi-user:
1. Log in as an administrator on Windows or as root on Linux.
2. Choose a root site for the install tree
3. Change the permissions on the root site so only you can write to the installed areas.
4. Create group access with only read and execute permission.
5. Install the Lotus Expeditor Client at this location. Select the multi-user configuration when you are prompted to do so during the installation.
6. Notify users to launch the desktop icon for the first time. When a user initially launches the desktop icon, a provisioning cycle is initiated for this user. The workspace and configuration space are initialized. The provisioning cycle will use the deploy manifest file to do this. For more information on customizing this manifest, see ["Customizing basic installation" on page 18].
   - The Lotus Expeditor Client is launched for the first time for the user. The user can use any of the provisioning modules provided to customize his or her environment.
7. Install additional applications or features, as needed. This will allow users to add them to their configuration. Users are not required to add the new features.

**Security**

The core install sites (${rcp.home}/eclipse, ${rcp.home}/rcp, and ${rcp.home}/shared) and any additional sites added as <root> sites must have the access rights controlled. Users should only be given read and execute access to these sites.

On Windows, the default access rights give any user change rights. This allows a user to add files to the Lotus Expeditor Client install tree. For a secure installation, it is recommended that you remove these change rights. If the change rights are not removed, a user can add an untrusted feature to a site by copying. At this point, any user could install it.

**First-time provisioning**

The first time that the multi-user environment is installed and setup, you, as administrator, select the level of base features to install. This sets the base level for all users. When a user starts Lotus Expeditor Client for the first time, the user's configuration, workspace, and feature set are provisioned to the same level as initially selected by the administrator. Provisioning guarantees that the version used for a feature is the same or higher.
This first-time provisioning happens once. From that point forward, the feature set is individualized for each user as needed.

**Provisioning and JXE cache initialization**

The JXE file cache, by default, is located in `${rcp.home}/rcp`. If this location is read-only by users, any time the administrator changes the features installed in a common location `${rcp.home}/eclipse`, `${rcp.home}/rcp`, and `${rcp.home}/shared`, the JXE file cache must be initialized. Having the JXE file cache fully initialized is critical for users runtime performance. After all provisioning changes are complete, exit the provisioning tool and run the following command:

```
${rcp.home}/rcplauncher -application com.ibm.jxesupport.initializer.initialize
```

When complete, the initializer application exits and the cache is fully initialized.

---

**Restricted Workbench Service**

Enable the Restricted Workbench Service to provide a restricted environment in which all Lotus Expeditor Client users are limited to the applications and operating system services that you, as administrator, have configured.

When the Restricted Workbench Service is enabled and a user without administration privileges logs on to the operating system, the Lotus Expeditor Client is launched and the Restricted Workbench Service is automatically enabled. For non administrative users, the Lotus Expeditor Client replaces Windows Explorer as the desktop shell on Windows systems. On Linux systems, the Lotus Expeditor Client replaces the GNOME Window Manager’s default session. The following restrictions are imposed on the Workbench:

- The Workbench alters its behavior, look, and feel, as follows:
  - No title bar
  - No sizing borders
  - Maximized to fill the screen
  - Pinned down in the Z-order such that no other windows can be drawn beneath it
  - Cannot be closed, re-sized, or minimized
  - The menu sequence **File → Exit** is removed from the Menu Bar

- The user is not able to gain access to the file system except through items contributed to the shared contribution areas of the Workbench (Cool Bar, Menu Bar, or Application Switcher Bar).

- The user is not able to gain access to any native applications except through items contributed to the Application Launcher.

- The user is not able to gain access to any operating system functions (for example, screen lockup, logoff, shutdown, change locale) except for the use of the **Alt + Tab** key combination to navigate between open native windows and through items contributed to the shared contribution areas of Workbench (Menu Bar and/or Eclipse Preference Pages)

- The user is blocked from performing the following key-stroke combinations:

**Windows:**

- **Alt + F4**, which closes the window with focus. This is only blocked on the Lotus Expeditor Client window; other windows can still be closed with **Alt + F4**.
- **Ctrl + Shift + Esc**, which opens the Windows Task Manager. Task Manager lets you stop processes (including Lotus Expeditor Client).
- Windows Logo Key + **L**, which locks the display.
- **Ctrl + Alt + Delete**, which displays the Windows Security dialog. The Window Security dialog is not blocked but all the buttons except for **Cancel** are disabled.
Linux: Alt + Ctrl + Backspace, which terminates the gnome session.

Alt + Ctrl + (F1 through F12), which switches between virtual terminals.

The Windows XP operating system introduced the notion of "Use Fast User Switching" to change credentials but not fully logout and exit all running applications. If these settings are enabled, the system security policy is also altered to provide a more streamline experience for the users. When running the Lotus Expeditor Client in Restricted Workbench mode, it is recommended you disable the "Welcome Screen" and "Use Fast User Switching" settings to further increase your administrator control over the system. For instance, if these settings are enabled, and a system screensaver is set, once the screensaver activates and then is deactivated, the user will be presented with a dialog allowing him or her to shutdown or logoff the system. Another modification relates to the Ctrl + Alt + Del key sequence, which presents a Windows Security dialog. If the IBM® Lotus Expeditor Restricted Workbench is installed, and the "Welcome Screen" and "Use Fast User Switching" settings are disabled, the only option available to the user on this dialog is Cancel. However, if the "Welcome Screen" and "Use Fast User Switching" options are enabled, the user will again have the ability to logoff or shutdown the system. When installing the Restricted Workbench, these settings are automatically disabled to provide a more secure environment.

To verify these settings, perform the following steps:
1. Click Start → Control Panel.
2. Click the User Accounts control panel.
3. Click Change the way users log on or off.
4. Modify the Welcome Screen and Use Fast User Switching options accordingly.
5. Click Apply Options.

Additionally, the default Administrator Windows XP account should not be used to install the IBM Lotus Expeditor Restricted Workbench. It is recommended that you create a secondary administrative user account and use this account to install the Restricted Workbench environment.

For more information, see "Installing with the Restricted Workbench Service" on page 11.

**Enterprise Management Agent Client**

You can enable your Lotus Expeditor Client workstations to be managed by an Lotus Expeditor Server. This is one way to centrally administer your Lotus Expeditor Client environment.

The Lotus Expeditor Server contains the Device Manager server component and provides basic platform management of the client along with additional features, such as support for iSync and MQe servers.

For more information, see "Managing using the Device Manager server" on page 181.

**Portal-administered client**

You can enable your Lotus Expeditor Client workstations to receive configuration information from a WebSphere® Portal server. This is one way to centrally administer your Lotus Expeditor Client environment.

The portlet container allows for many different client side architectures with the composite application infrastructure. There are two primary ways applications can run on the client:

- A standard widget toolkit (SWT)-based application aggregated and deployed by Portal. This application is composed of SWT-based views and is placed on the Portal with the various view components wired together with the Portal wiring tool.
- Locally running portlets. Properly bundled portlets can be run locally and off-line in the Lotus Expeditor Client. This ability, mixed with Property Broker, allow for the locally running portlets to
inter-communicate with each other. These disparate components can then communicate with each other using the declarative wire expressed in the composite application XML.

Additionally, a third way of encapsulating Web applications with other components is by wrapping the Managed Browser in a managed browser SWT view that exposes property broker actions and properties. This has the same effect as a traditionally wired experience of any portlet or view driving another component (in this case a browser taking in URLs).

For more information, see ["Managing Rich Clients using the Portal server" on page 205](#).

### Network Client Install

Install the Network Client Installer so that users can remotely install the Lotus Expeditor Client, install the administrative components needed on your Portal server, and manage the Lotus Expeditor Client.

See ["Installing with the Network Client Installer" on page 29](#) for more information.

### Understanding the client file layout

When the Lotus Expeditor Client 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.

```xml
<installation directory>/
    _uninst/   - Files required for uninstalling the product Linux only
        eclipse/ - Platform components
            .eclipseproduct
            features/
                links/
                    plugins/
            ITLM/     - For license management
                plugins/
                _uninst
                deploy/
                    jxes<version> - Platform launcher
                        rcplauncher.exe
                        rcplauncher.properties - Platform launcher configuration file
                        startcollector.bat - IBM Support Assistant log collector utility (Windows)
                        startcollector.sh - IBM Support Assistant log collector utility (Linux)
            eclipse/ - Platform components
                .eclipseproduct
                features/
                    plugins/
                shared/ - Site to contain apps shared across multiple configs
                    eclipse/
                        .eclipseextension
                            features/
                                plugins/
            license/ - Product licenses in multiple languages
```

**Note:** The _uninst directories appear when using an ISMP installer on Linux.

**Related tasks**

["Updating the rcplauncher.properties file" on page 89](#)

This section describes how you can update one or more elements of a user’s Lotus Expeditor Client platform by modifying the rcplauncher.properties file.

**Related information**

["Configuring the platform launcher" on page 44](#)

You can configure additional arguments to use when the user launches the Lotus Expeditor Client. For example, you can specify that a console is displayed when launching the client.
The Lotus Expeditor Client Workbench leverages Eclipse products and Eclipse themes for branding. You should create your own branding plug-in with your product information instead of changing the one shipped with the Lotus Expeditor Client. Create your own branding feature and deploy it as part of your initial install.

Single sign-on with the operating system

Configuring the IBM Lotus Expeditor platform to provide single sign-on (SSO) with the operating system enables a user, who has successfully logged into the operating system, to gain access to resources protected by the IBM Lotus Expeditor platform without ever being prompted for a password.

The IBM Lotus Expeditor platform stores security-related information, such as authentication credentials (passwords), keys, and certificates in the local keystore. When SSO with the operating system is enabled, the IBM Lotus Expeditor platform stores the password that is used to unlock the credential store in the operating system’s native credential store.

Managed settings

The managed settings framework allows you control over the runtime behavior of client applications by letting you set the values of the settings that the applications are reading out of the Eclipse preference store.

If you also designate these settings as read-only, any changes that the client applications make to the settings will either be prevented or be subsequently overwritten with your values, depending on how the client accesses the settings. If the client program knows about managed settings and accesses them through the ManagedSettings Eclipse preference scope, any changes to read-only settings are prevented. For other plug-ins, which access the settings in the traditional way, changes can occur but are later overwritten with your values. Updates are run regularly to add any of your new settings or settings changes to the preference store on the client.

The managed settings framework can retrieve policy, preferences, or any other name and value pairs from any back-end system for which there exists a managed settings provider. Providers contain the knowledge of the specific back-end system. Lotus Expeditor Client includes providers to retrieve settings both from Portal Policy Manager and from an XML file residing on an HTTP server. Lotus Notes® 8.0 includes a provider to retrieve policies from Domino® using the Notes® client. If there is no provider available for your back-end system, you may write your own. See "Adding a managed settings provider" on page 135 for more information.

Note: Managed settings will not work when using Portal 6.0.1. Portal 6.0 or Portal 6.0.1.1 must be used.

Related tasks

"Specifying platform branding and theme" on page 115

"Configuring Managed Settings" on page 70
Installing and launching the Lotus Expeditor Client

This section describes how to install and launch Lotus Expeditor Client onto client machines.

There are several installation methods available. Before you install Lotus Expeditor Client, ensure that each machine on which you are installing meets the client prerequisites.

Installing the Lotus Expeditor Client on the desktop

This section describes how to install Lotus Expeditor Client on the desktop.

Client prerequisites

Ensure that each machine on which you plan to install the Lotus Expeditor Client meets these requirements.

Client for Desktop

**Table 1.**

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Minimum software requirements</th>
<th>Minimum hardware requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>• Microsoft® Windows XP Professional Service Pack 1&lt;br&gt;• Microsoft Windows XP Home Edition Service Pack 1&lt;br&gt;• Microsoft Windows XP Tablet PC Edition 2005&lt;br&gt;• Microsoft Windows 2000 Professional Service Pack 4&lt;br&gt;• Windows Vista* (see &quot;Understanding Microsoft Windows Vista installations&quot; on page 21 for more information)</td>
<td>• x86 processors capable of supporting the supported operating systems.&lt;br&gt;• 512MB RAM&lt;br&gt;• 200 MB free disk space&lt;br&gt;• Display resolution: Super video graphics array/adapter (SVGA) 1024 x 768 display&lt;br&gt;• High contrast mode: 1152 x 768 display, only default font settings (10pt Sans)</td>
</tr>
<tr>
<td>Linux</td>
<td>• Red Hat Enterprise Linux 4.0 WS with GTK support - Update 4&lt;br&gt;• Red Hat Enterprise Linux 5.0 WS with GTK support&lt;br&gt;• SUSE Linux Enterprise Desktop (SLED 10)</td>
<td>• x86 processors capable of supporting the supported operating systems.&lt;br&gt;• 512MB RAM&lt;br&gt;• 200 MB free disk space&lt;br&gt;• Display resolution: Super video graphics array/adapter (SVGA) 1024 x 768 display&lt;br&gt;• High contrast mode: 1152 x 768 display, only default font settings (10pt Sans)</td>
</tr>
</tbody>
</table>

**Note:** Hebrew and Arabic languages are not supported on Linux platforms. In addition, bi-directional text (-dir rtl) is not supported.
IBM Lotus Expeditor requires Mozilla Firefox or Mozilla Runtime Environment on Linux

To use IBM Lotus Expeditor on Linux, you must first have the Mozilla browser installed. Select from the following:
- Mozilla Firefox 1.5.x.x (GTK2 build)
- Mozilla SeaMonkey 1.0.x.x (GTK2 build)
- Mozilla Runtime Environment (xulrunner) 1.8.0.x (GTK2 build)

Install the Mozilla browser or xulrunner with a compliant version from your installation media. Visit the Linux update site or the Mozilla Web site [http://www.mozilla.org] to download and install these programs.

Verifying the Mozilla environment on a Linux system

Before launching IBM Lotus Expeditor, follow these steps to verify you meet the prerequisites for Linux:

### SUSE Linux Enterprise Desktop 10
The default installed xulrunner1.8.0.1 can be used. The path is /usr/lib/xulrunner1.8.0.1. The default installed Firefox 1.5.0.4 cannot be used.

You can find the prerequisite packages on your SLED 10 Service pack CDs or from the Linux update site.

### Red Hat Enterprise Linux 4.0 WS with GTK support - Update 4
The default installed Firefox can be used. The path is /usr/lib/firefox1.5.0.3.

Install Firefox or xulrunner from the Linux update site or the Mozilla Web site [http://www.mozilla.org].

### Red Hat Enterprise Linux 5.0
The default installed Firefox 1.5.0.9 can be used. The path is /usr/lib/firefox1.5.0.9.

You can find it on your Red Hat Enterprise Linux 5.0 CDs, from the Linux update site, or the Mozilla Web site [http://www.mozilla.org].

Follow these steps to verify which Mozilla application is installed on your Linux system:

### Mozilla Runtime Environment (xulrunner)
1. Open a terminal and enter `rpm -qa | grep xulrunner` to query installed packages.
2. If you see returns like `mozilla-xulrunner-1.8.0.x-xx`, the Mozilla Runtime Environment (xulrunner) is installed.

### Mozilla Firefox
1. Open a terminal and enter `rpm -qa | grep mozilla-firefox` to query installed packages.
2. If you see returns like `mozilla-firefox-1.5.xx-xx`, Mozilla Firefox is installed.

### Mozilla SeaMonkey (Mozilla Browser Suite)
1. Open a terminal and enter `rpm -qa | grep mozilla-seamonkey` to query installed packages.
2. If you see returns like `mozilla-seamonkey-xx-xx`, a compliant Mozilla SeaMonkey is installed.

Related information

- "Installing from a CD" on page 9
  This section provides the steps involved in installing Lotus Expeditor Client from a CD.
- "Invoking a silent installation" on page 12
  You can set up the Lotus Expeditor Client installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file on Linux or a response file on Windows, you can customize the silent installation to automatically select panel options and installation properties.
Installing from a CD

This section provides the steps involved in installing Lotus Expeditor Client from a CD.

Before you install the Lotus Expeditor Client, 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 200 MB free.

Use the following steps to install Lotus Expeditor Client from a CD.
1. Insert the Lotus Expeditor Client CD into the CD-ROM drive.
2. Launch the appropriate installer executable for your system. The installer is located in the insta11/ directory of the CD.
   a. On a Windows system, run the setupwin32.exe file.
   b. On a Linux system, run the setuplinux.sh file.

Notes:
   • If a File Download dialog displays, click Open to run the installer.
   • You can run the installer directly from the insta11/ directory on the installation CD instead of the browser.
3. The Lotus Expeditor Client Installer guides you through the rest of the installation:
   a. At the installer welcome page, click Next.
   b. Read the license. To accept the license, select I accept the terms in the license agreement and click Next.
   c. On Windows, if a previous version is detected, you are prompted to upgrade. Click Next to continue and proceed to step 3b on page 10 If the current version is detected to already be installed, you are prompted to uninstall.
      During installation, the installer checks for Lotus Expeditor Client 6.1. If it is found, your settings are upgraded and the installation continues. If you have 6.0 installed, you must first install and migrate to 6.1 using the 6.1 installer and then upgrade to 6.1.x.
   d. Enter the destination where you want to install the Lotus Expeditor Client and click Next.

Note: You can only install the Lotus Expeditor Client from or to a directory containing valid ASCII characters.

Additionally, the default destination may not be accessible by the current user. In this case, change the location to a user-accessible destination.

Examples:

Windows
   C:\Documents and Settings\user99\IBM\Lotus\Expeditor

Linux
   /home/user99/IBM/Lotus/Expeditor

e. On Linux, if a previous version is detected at the destination specified in step 3d, you are prompted to upgrade. Click Next to continue and proceed to step 3b on page 10. If the current version is detected to already be installed, you are prompted to uninstall.
f. Select the type of installation (Single User or Multi-User) and click Next. For more information, see “Single user configuration” on page 1 and “Multi-user configuration” on page 1.
g. 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 181.
h. If you selected to configure the Enterprise Management Agent, proceed to the next step, otherwise proceed to step 3k on page 10.
i. Provide settings for the Enterprise Management Agent and click **Next**.

j. To set up the product to be administered by a Portal server, select **Yes** and enter a Portal server URL, otherwise select **No**. Click **Next** to continue.


k. If you selected a multi-user install configuration, proceed to the next step, otherwise proceed to step **3n**.

l. To allow users the option of launching the product without entering a password, choose a setting and click **Next**.

m. If you selected to add password configuration capability, proceed to the next step, otherwise proceed to step **3o**.

n. If you are presented with a panel asking to enable a password prompt at start up, choose a setting and click **Next**.

o. To launch the platform after installation is successful, select **Yes**, otherwise select **No**. Click **Next** to continue.

p. Before the actual installation begins, verify the installation summary information is correct. Click **Next** to begin the installation.

q. After the installation is complete, click **Finish**.

**Related information**

"Client prerequisites" on page 7

Ensure that each machine on which you plan to install the Lotus Expeditor Client meets these requirements.

"Invoking a silent installation" on page 12

You can set up the Lotus Expeditor Client installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file on Linux or a response file on Windows, you can customize the silent installation to automatically select panel options and installation properties.

**Installing from a shared network location**

This section provides the steps involved in installing Lotus Expeditor Client from a shared network location.

**Use the following steps to install Lotus Expeditor Client from a shared network location.**

1. Map to the shared folder.
2. Continue with step 3 described in "Installing from a CD" on page 9

**Note:** If installing on Windows Vista, you must run the install as an administrator:

1. In Windows Explorer, right-click `setupwin32.exe` and select **Run as Administrator**.
2. Follow the prompts to complete the installation.
3. Launch IBM Lotus Expeditor as the standard user using `<install location>cpcplauncher.exe`. This creates a new workspace and desktop icon for the standard user.

For network share install to complete successfully on Windows Vista, you might need to make adjustments to the Local security policy for UAC or disable it.

**Related information**

"Client prerequisites" on page 7

Ensure that each machine on which you plan to install the Lotus Expeditor Client meets these requirements.

"Invoking a silent installation" on page 12

You can set up the Lotus Expeditor Client installation program to run silently if you do not want to
interact with the installation wizard. Additionally, using an options file on Linux or a response file on Windows, you can customize the silent installation to automatically select panel options and installation properties.

**Installing with the Restricted Workbench Service**

Enable the Restricted Workbench Service to provide a restricted environment in which all Lotus Lotus Expeditor Client users are limited to the applications and operating system services that you, as administrator, have configured.

**6.1.2** The Restricted Workbench Service can only be enabled during initial installation; it cannot be added to an existing installation. The Expeditor Restricted Workbench client does not support uninstallation. Running the uninstall program may appear to function properly, but the system will encounter problems when logging in as a previously restricted user afterwards. It is necessary to reinstall the base operating system instead of attempting to remove the Expeditor Restricted Workbench.

**Notes:**

- **6.1.2** If performing an upgrade of a previously-installed Windows XP Restricted Workbench environment, it is necessary to ensure Windows Installer 3.1 or greater is already installed and configured, or the system will become unusable.
- Due to a known limitation of Java™, it is necessary to execute the following commands as the root user on Linux machines, after the initial IBM Lotus Expeditor Restricted Workbench installation:
  1. Exit IBM Lotus Expeditor if it is running
  2. Specify the following:

     **6.1.1**
     
     ```
     cd /opt/ibm/Lotus/Expeditor/rcp/eclipse/plugins/com.ibm.rcp.lockdown.gtk_6.1.1.<date>/install
     ```

     or

     **6.1.2**
     
     ```
     cd /opt/ibm/Lotus/Expeditor/rcp/eclipse/plugins/com.ibm.rcp.lockdown.gtk_6.1.2.<date>/install
     ```

  3. If the directory uninstall/restricted exists, specify `rm -rf uninstall/restricted`
  4. Specify `./lockdown.sh install`

The Restricted Workbench Service is available in multi-user installation environments only. To enable the Restricted Workbench Service during product installation, ensure that the installation manifest (whether using the provided version or creating your own custom manifest file) contains the following information:

```xml
<feature
  id="com.ibm.rcp.lockdown.feature"
  version="<feature_version>"
  match="compatible"
  action="install"
  shared=false"
/>```

See "Customizing basic installation" on page 18 for more information on installing manifest files.

**Related concepts**

"Accessing the operating system" on page 12

As an administrator, you can access the operating system while the Restricted Workbench Service is installed.
**Accessing the operating system**

As an administrator, you can access the operating system while the Restricted Workbench Service is installed.

To access the unrestricted operating system, for example the Explorer shell in Windows XP or Gnome shell in Linux, follow these steps:

1. Log on to the operating system as an administrator.
2. The default operating system window shell is displayed.

You are logged in to the unrestricted operating system.

**Changing Home Portal Account settings for users in the Restricted Workbench environment**

You can change the Home Portal Account settings for users in the Restricted Workbench environment.

To change the Home Portal Account server address or username and password for users in the Restricted Workbench environment, a user with administrative privileges must launch the Lotus Expeditor Client using the `-data <users workspace directory>` parameter and then access the Home Portal Account preferences panel:

1. If you are using Linux, specify `chown -R root <users workspace>` to modify owner permissions of the `<users workspace>` directory containing logs, applications, and `.config` directories (for example, `/home/myuser/IBM/RCP/<ID number>/myuser`).
2. Launch the Lotus Expeditor Client with the following command line parameters: `-data <users workspace>`
3. Click File > Preferences then click the Home Portal Account in the left panel.
4. Make the necessary modifications to the settings displayed and click OK to apply the changes.
5. Allow the Lotus Expeditor Client to restart and make sure the new applications are available.
6. Exit the Lotus Expeditor Client.
7. If you are using Linux, specify `chown -R <user> <users workspace>` to restore owner permissions of the `<users workspace>` directory.
8. Log out of the operating system administrative user.
9. Log in to the operating system as the Restricted Workbench user that will automatically start the Lotus Expeditor Client. The applications are now available.

**Invoking a silent installation**

You can set up the Lotus Expeditor Client installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file on Linux or a response file on Windows, you can customize the silent installation to automatically select panel options and installation properties.

Note: The installer executable is located in the `install/` directory of the CD.

1. On a Windows system, run the `setupwin32.exe` file.
2. On a Linux system, run the `setuplinux.sh` file.

To install silently with default options:

1. From a command line, change to the `install/` directory on the CD.
2. Run the following command:

   On Windows:
   ```bash
   <setup executable> /s /v"/qn LICENSEACCEPTED=true"
   ```
   For example, `setupwin32.exe /s /v"/qn LICENSEACCEPTED=true"`.

   On Linux:
   ```bash
   <setup executable> -silent -V licenseAccepted=true
   ```
For example, `setuplinux.sh -silent -V licenseAccepted=true`.

To install silently with custom options:

**Note:** Templates of response and option files are located in the `<Expeditor_install_pack>\desktop\install` directory.

1. Create an options file using the steps provided in "Using options files" on page 14 or create a response file on Windows using the steps provided in "Using response files" on page 16.
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:
   - On Windows:  `<setup executable> /s /v"/qn USEINI=<fully qualified path to response file>"`
     
   For example, `setupwin32.exe /s /v"/qn USEINI=c:\temp\options.txt"`
   - On Linux:  `<installer executable> -silent -options option file`
     
   For example, `setuplinux.sh -silent -options /tmp/options.txt`.

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

   - On Windows:  `LICENSEACCEPTED=true`
   - On Linux:  `-V licenseAccepted=true`

   See the examples above.

**Related information**

"Client prerequisites" on page 7

Ensure that each machine on which you plan to install the Lotus Expeditor Client meets these requirements.

"Installing from a CD" on page 9

This section provides the steps involved in installing Lotus Expeditor Client from a CD.

**Removing the Lotus Expeditor Client**

Use the steps in this section to uninstall the Lotus Expeditor Client.

On Windows:

1. Launch **Add/Remove Programs** by selecting **Settings** → **Control Panel** → **Add/Remove Programs**
2. Select **Lotus Expeditor Client** and click **Change** (to uninstall using the uninstall wizard prompts) or **Remove** (to uninstall without any uninstall wizard prompts).

   **6.1.1** If you click **Change**, you will be presented with an uninstall wizard that will allow you to choose if you would like to remove user data. If you click **Remove**, no uninstall wizard will appear, and user data will be preserved after the uninstallation.

   **6.1.2** In version 6.1.2, the uninstall wizard does not provide the option to remove user data. You must manually remove it. Refer to "Removing user-specific data" on page 14.

On Linux, run `<install dir>/_uninst/uninstaller.sh`.
User data is stored in the workspace. See “Overriding the workspace directory location” on page 20 for more information.

### Silently uninstalling the Lotus Expeditor Client on Windows

#### 6.1.1 To silently remove Lotus Expeditor version 6.1.1 with default options:

1. From a command line, change to the /desktop/install/ directory on the CD.
2. Run the following command: `<setup executable> /x /s /v"/qn"

#### 6.1.1 To uninstall silently with custom options:

1. Use a response file created during install using the steps provided in “Using response files on page 16.
2. Modify the response file with the customized values for your uninstallation.
3. From a command line, change to the /desktop/install/ directory on the installation CD.
4. Run the following command: `<setup executable> /x /s /v"/qn USEINI= fully qualified path to response file"`
   
   For example, setupwin32.exe /x /s /v"/qn USEINI=c:\temp\response.txt".

#### 6.1.2 To silently remove Lotus Expeditor Version 6.1.2 run the following command:

"<client destination>\uninstall.bat" -silent

### Removing user-specific data

In Lotus Expeditor version 6.1.2, after removing Lotus Expeditor from a user’s system, the user-specific data (the workspace) is left behind. If you do not want the user-specific data on the system you must manually remove it.

**Note:** On multi user installations, you must complete these steps for each account.

User-specific data (the workspace) from a previous installation of Lotus Expeditor can be reused if Lotus Expeditor Client is reinstalled on the system. To remove the data:

Delete the workspace location. The default locations are:

1. Windows:Documents and Settings/<user ID>/Application Data/Lotus/XPD.
2. Linux: /home/<user ID>/Lotus/XPD /<user ID>.

If you have changed this from the default location, then you should delete the overridden location.

### Using options files

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

The options you specify in the options file are processed after the processing of any options that you enter directly on the command line. You can generate an options file template 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. The file is named uninstaller.sh.
- When running the options-template option from CD, ensure that you specify a writable location for the output file (that is, use a full path to a hard disk location because you cannot write to the CD).

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.

Options available for install

Note: All option values should be enclosed in double-quotes (for example, `-V
installWelcomeAllowPanel=true`).

The following options control the execution of the install panels. Set the options to true (or false) to allow (or disallow) panels from executing. By default, all panels are allowed.

- `-V installWelcomeAllowPanel`
- `-V licenseAllowPanel`
- `-V destinationAllowPanel`
- `-V installConfigurationAllowPanel`
- `-V useDeviceManagementAllowPanel`
- `-V portalConfigurationAllowPanel`
- `-V securityConfigureAllowPanel`
- `-V securityPromptAllowPanel`
- `-V launchPlatformAllowPanel`
- `-V preInstallSummaryAllowPanel`
- `-V provisioningStatusAllowPanel`
- `-V postInstallSummaryAllowPanel`

Use `-V licenseAccepted` to control whether the license has been accepted for the install (true or false). The default value is false.

Use `-V installConfiguration` to determine the install configuration. Valid values are user and multiuser. The default value is user.

Use `-V useDeviceManagement` to determine if the Enterprise Management Agent is utilized (true or false). The default value is false.

The following options are used to configure the Enterprise Management Agent user name, password and server URL.

- `-V deviceManagementUserName`
- `-V deviceManagementUserPassword`
- `-V deviceManagementServerURL`

The following options configure the use of Portal management (true or false) as well as the Portal server URL.

- `-V usePortalManagement`
- `-V portalConfigurationServerURL`
Use -V securityAllowConfiguration to allow users the ability to configure whether they want to allow a password prompt (true or false). The default value is true.

Use -V securityPasswordPrompt to enable the platform to prompt for a password at start up (true or false). The default value is true.

Use -V launchPlatformOnExit to specify whether to launch the platform when the install completes successfully or not (true or false). The default value is true.

Use -P installLocation to specify the install location of the product. Specify a valid directory into which the product should be installed. The default is /opt/IBM/Expeditor.

**Options available for uninstall**

**Note:** All option values should be enclosed in double-quotes (for example, -V uninstallWelcomeAllowPanel="true").

The following options control the execution of the uninstall panels. Set the options to true (or false) to allow (or disallow) panels from executing. By default, all panels are allowed.

- -V uninstallWelcomeAllowPanel
- -V preUninstallSummaryAllowPanel
- -V provisioningStatusAllowPanel
- -V postUninstallSummaryAllowPanel

[ 6.1.1 ] An additional option is available if you are using Lotus Expeditor version 6.1.1:

- -V removeUserDataAllowPanel

Use -V removeUserData to determine if user-specific data should be removed during uninstallation (true or false). The default value is false.

**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**

**Invoking a silent installation** on page 12

You can set up the Lotus Expeditor Client installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file on Linux or a response file on Windows, you can customize the silent installation to automatically select panel options and installation properties.

**Client prerequisites** on page 7

Ensure that each machine on which you plan to install the Lotus Expeditor Client meets these requirements.

**Invoking a silent installation** on page 12

You can set up the Lotus Expeditor Client installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file on Linux or a response file on Windows, you can customize the silent installation to automatically select panel options and installation properties.

**Using response files**

On Windows, you use response files to pass command line properties to an installation as you would normally specify these properties on the command line. These properties represent the response to the installation wizard panels, and set installation properties accordingly.
Saving a response file

To designate the response file (or initialization file) to be created run the following command:
<setup executable> /v"SAVEINI=c:\myfile.ini"

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

Note:
- SAVEINI="" suppresses creation of response file.
- By default, .ini file is not removed when product is uninstalled.

Using the response file during installation

After saving a response file (or initialization file) on installation, run the following command to designate the initialization file to be used for an installation:
<setup executable> /v"USEINI=c:\myfile.ini"

Removing a response file during uninstallation 6.1.1

On Lotus Expeditor version 6.1.1, to designate that the initialization file should be deleted during uninstall run the following command:
<setup executable> /v"REMOVEINI=1"

Creating an initialization file only

To create the initialization file but not install the product run the following command:
<setup executable> /v"SAVEINI=c:\myfile.ini ONLYINI=1"

Note: If ONLYINI=1 is specified, a SetupInterrupted dialog follows the ReadyToInstall dialog and a message is written to the log file.

Options available for initial install

Use LICENSEACCEPTED to control whether the license has been accepted for the install (true or false). The default value is false.

Use INSTALLCONFIGURATION to determine the install configuration. Valid values are user and multiuser. The default value is user.

Use USEDEVICEMANAGEMENT to determine if the Enterprise Management Agent is utilized (true or false). The default value is false.

The following options are used to configure the Enterprise Management Agent user name, password and server URL:
- DEVICEMANAGEMENTUSERNAME
- DEVICEMANAGEMENTUSERPASSWORD
- DEVICEMANAGEMENTSERVERURL

The following options configure the use of Portal management (true or false) as well as the Portal server URL:
- USEPORTALMANAGEMENT
- PORTALCONFIGURATIONSERVERURL
Use SECURITYALLOWCONFIGURATION to allow users the ability to configure whether they want to allow a password prompt (true or false). The default value is true.

Use SECURITYPASSWORDPROMPT to enable the platform to prompt for a password at start up (true or false). The default value is true.

Use LAUNCHPLATFORMONEXIT to specify whether to launch the platform when the install completes successfully or not (true or false). The default value is true.

Use INSTALLDIR to specify the install location of the product. Specify a valid directory into which the product is be installed. The defaults is C:\Program Files\IBM\Lotus\Expeditor.

**Options available for upgrade 6.1.2**

Use LICENSEACCEPTED to control whether the license has been accepted for the install (true or false). The default value is false.

Use LAUNCHPLATFORMONEXIT to specify whether to launch the platform when the install completes successfully or not (true or false). The default value is true.

**Note:** Other initial install options should not be used during upgrade.

**Related information**

- [Invoking a silent installation](#) on page 12
  You can set up the Lotus Expeditor Client installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file on Linux or a response file on Windows, you can customize the silent installation to automatically select panel options and installation properties.

- [Client prerequisites](#) on page 7
  Ensure that each machine on which you plan to install the Lotus Expeditor Client meets these requirements.

- [Invoking a silent installation](#) on page 12
  You can set up the Lotus Expeditor Client installation program to run silently if you do not want to interact with the installation wizard. Additionally, using an options file on Linux or a response file on Windows, you can customize the silent installation to automatically select panel options and installation properties.

**Customizing basic installation**

The Lotus Expeditor Client installation process consists of two steps: installation of core platform files and provisioning of additional components. Depending on the desired install configuration (single or multi-user), differing sets of components are provisioned to the product from update sites.

**6.1.1** The set of components to be provisioned is defined within provisioning manifests, 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 provisioning manifests must be named user.xml or multiuser.xml (corresponding to the available install configurations).

**6.1.2** The set of components to be provisioned is defined within a provisioning manifest, which is processed by the installer and platform provisioning system. The file must be named install.xml and be located in a deploy/ directory located in the same location as the installer executable.

The Lotus Expeditor Client CD includes an provisioning manifest for each of these install configurations. 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 install configurations include a customized set of components.

**Note:** Modifying these provisioning manifests from their shipped states could result in an invalid platform installation.

### 6.1.1 During the installation process, the provisioning manifest corresponding to the chosen install configuration is processed (replacing any supported variables) and copied to `<product destination>/rcp/deploy/install.xml`.

### 6.1.2 During the installation process, the provisioning manifest is processed (replacing any supported variables) and copied to `<product destination>/rcp/deploy/install.xml`.

The location of this file is then passed to the platform provisioning component to begin the provisioning process.

For more information on using the provisioning manifest and its syntax, see *"Using the provisioning manifest" on page 160*.

**Overriding the list of update sites for provisioning**

You can override the list of update sites used to search for features to install.

By default, the IBM Lotus Expeditor installer accesses an update site in a location relative to the setup executable to provision the platform components. You can override the list from the command line:

1. From the command line, change to the install/ directory on the CD.
2. On Linux, enter the following command using the `updateSiteList` variable as follows:
   ```
   setuplinux.sh -V updateSiteList="<site 1>|<site 2>|...|<site n>"
   ```

   On Windows, enter the following command using the `UPDATESITELIST` property as follows:
   ```
   setupwin32.exe /v"UPDATESITELIST="<site 1>|<site 2>|...|<site n>"
   ```

   In either case, each update site specified must be a valid URL. Also, if the site specified contains the text `${installer.root}`, that text will be replaced by a file URL value that contains the location from where the installer launched.

**Installing the Lotus Expeditor Client for shared launching from a network drive**

The Lotus Expeditor Client can be installed onto a drive that can be shared as a network drive. This enables a single install location on the network to be shared among many users.

The installation must adhere to the following restrictions:

- Select the **Multi-User install** option. This sets configuration parameters to favorable settings for a shared install.
- The install can be used only by clients running on the same operating system family as the install application. In other words, if you are using Windows to install the Lotus Expeditor Client, then the installation can be used only by users on Windows systems. If you are using Linux to install the Lotus Expeditor Client, then the installation can be used only by users on Linux systems.
- The install can be made either to a local drive that is later shared with other users, or it can be installed to a shared drive. On Windows, when installing to a shared drive, the shared drive must be mounted as a drive letter. UNC mounted destinations are not supported.

Installing to a shared drive mounted with the Linux Samba client is not supported. If you are using Linux and attempt to use the Samba client to mount a shared folder and install into that shared folder,
you will receive errors indicating that files could not be renamed. This is a known issue with this configuration. An alternative would be to use NFS for mounting shared folders.

- Use the policies applicable to the multi-user configuration for deploying additional features to the shared sites. See "Multi-user configuration" on page for more information.

To enable use of a network shared installation, the configuration related properties files do not contain absolute directory and file location references, but use the following variables that are resolved and replaced during the launch process:

\$\{rcp.data\}

    Refers to the workspace directory. See "Overriding the workspace directory location" for more information.

\$\{rcp.home\}

    Refers to the platform install location. The value for rcp.home is set by the launcher to be the parent directory of the directory containing the launcher.

    On Windows, the default install location is C:\Program Files\IBM\Lotus\Expeditor. The Expeditor directory contains a directory rcp, in which resides the launcher, rcplauncher.exe. rcp.home is set to C:\Program Files\IBM\Lotus\Expeditor.

    On Linux, the default install location is /opt/ibm/lotus/Expeditor. The Expeditor directory contains a directory rcp, in which resides the launcher, rcplauncher. rcp.home is set to the /opt/ibm/lotus/Expeditor directory.

The default install sites – eclipse, rcp, and shared – reside under the rcp.home directory. Install sites cannot be added unless they are also referenced as children of the rcp.home root.

**Overriding the workspace directory location**

You can override the workspace directory location.

On Windows, the default workspace location is <APPDATA>\Lotus\XPD.

On Linux, the default workspace location is <HOME>/Lotus/XPD where

<APPDATA> and <HOME> are system environment variables. In default Windows installations, the workspace directory would resolve to Documents and Settings\<userid>\Application Data\Lotus\XPD.

In default Linux installations, it would resolve to /home/<userid>/Lotus/XPD.

**Note:** On installations upgraded from IBM Lotus Expeditor 6.1.0, the workspace location will be preserved from the 6.1.0 installation. On both systems, this location is <user home directory>/IBM/RCP/<rcp_install_ID>/<user name>.

When specifying the new directory location, the location must be accessible to all users that will be using the workspace. For a multi-user configuration, where each user will have their own unique workspace, the workspace directory location should be specified containing environment variables that will be replaced on launch and will therefore resolve to a unique location for each user. If in a multi-user configuration, a common workspace will be serially shared by all users logging onto the system. The workspace directory must be in a location accessible to all users, with full permissions on the common location.

Use the following command line options to override the workspace directory location:

- On Linux, enter the command using the rcpData variable as follows:

  setuplinux.sh -V rcpData="<desired workspace location>"

  If not specified on Linux, $\{env.HOME\}/Lotus/XPD is used.

  Example:
setuplinux.sh -V rcpData=${HOME}/Lotus/XPD

- On Windows, enter the command using the rcpData variable as follows:
  setupwin32.exe /v"RCPDATA=""<desired workspace location>"

  If not specified on Windows, ${APPDATA}/Lotus/XPD is used.

  **Note:** If the command line override is used, logging is not automatically enabled. Add /i\wemo 
  "<install log location>" to enable logging.

  Example:
  setupwin32.exe /v"/i\wemo /v"RCPDATA="<install log location>"\" RCPDATA="${APPDATA}/Lotus/XPD\""

Features that were installed into one workspace cannot be uninstalled from a different workspace. For example, when you initially launch IBM Lotus Expeditor using the default workspace, a set of features are installed and provisioned for you. If you later launch with a different workspace using the rcpData launch option, the features that were installed into the default workspace cannot be uninstalled from the new workspace. If you attempt to uninstall these features, they will not be uninstalled and errors will be written to the error log.

**Locating the Lotus Expeditor Client launcher program**

IBM Lotus Expeditor creates registry keys on Windows and files on Linux to allow for the location of the Lotus Expeditor Client launcher.

On Windows, the registry path depends on the install attributes. If an administrator has installed the Lotus Expeditor platform, the registry path is HKEY_LOCAL_MACHINE\SOFTWARE\IBM\Lotus\Expeditor. Otherwise, the path is HKEY_CURRENT_USER\SOFTWARE\IBM\Lotus\Expeditor. Within this registry folder, there will be another key created based on the product.name of the installed product. Within this new key, the string value launcher will be created to contain the path of the rcplauncher.exe program. When searching the registry, first check HKEY_CURRENT_USER. If the keys are not found, then check HKEY_LOCAL_MACHINE.

On Linux, a configuration file will be created to contain the location of the launcher. If an administrator has installed the IBM Lotus Expeditor platform, the configuration file will be created in /etc/ibm/lotus/expeditor directory. Otherwise, the path is $HOME/user/.expeditor. Within the directory, a file named .cfg will be created to contain the path to the rcplauncher program. As an example, the file contents would be the following line:

`launcher=/opt/ibm/lotus/Expeditor/rcp/rcplauncher`

IBM Lotus Expeditor uses the product.name **Expeditor** in its branding feature to create the registry entries as described above. Installations that have replaced the IBM Lotus Expeditor branding feature with another branding feature might change the product.name, and as a result, the file name containing the launcher information will be different.

**Understanding Microsoft Windows Vista installations**

Read this section for an understanding of User Account Control and the security model associated with Microsoft Windows Vista.

**Microsoft Windows Vista User Access Control**

Microsoft Windows Vista implements a new security model referred to as User Account Control (UAC), which when enabled limits access to protected directories and sections of the registry. These limits are imposed on standard users and members of the Administrator group. Members of the Administrator group are prompted for access each time they attempt to access the resource. Likewise, users must provide the password of a member of the Administrators group each time they want to access the...
resource. An additional limitation in both scenarios is the dialog prompting for consent or the password has a two-minute timeout after which it will disappear and interrupt software installations or deny access to the requested resource.

UAC also implements resource virtualization for the protected directories and sections of the registry. An example of resource virtualization is a standard user trying to create a file in C: \ Program Files \ <MyApp> \ <MyFiles> will not receive an access error; rather the file will be created in C: \ Users \ <user> \ AppData \ Local \ VirtualStore \ Program Files \ <MyApp> \ <MyFiles>. If UAC is disabled, resource virtualization is disabled.

IBM Lotus Expeditor can be installed with UAC enabled or disabled, though it is recommended to keep User Account Control enabled to help secure the system.

**Administering the platform on Windows Vista**

With the advent of the new Microsoft security model in Vista, the platform is inherently more secure as all users are limited to standard user access rights. Administrative users can request elevated privileges when they need to perform administrative functions. For IBM Lotus Expeditor, you must launch the platform requesting the elevated privileges when installing additional features, NLS language support, or making other configuration changes which might affect all users. This will allow proper creation of new application icons on the Windows Desktop or in the Windows Start Menu and proper creation of global registry keys. The platform should not be run with elevated privileges excessively as some functions such as launching external browsers will not behave as expected. To run the platform with elevated privileges:

1. Log in to Vista as a user part of the Administrators group
2. Right-click on the IBM Lotus Expeditor icon (typically on the desktop or in the Start Menu)
3. Select **Run as administrator**
4. Respond to the Vista prompt for consent to run the program with elevated privileges

---

**Launching the Lotus Expeditor Client on the desktop**

This topic describes how to start the Lotus Expeditor Client, how the daemon process works, and how an existing process is joined if a new personality is requested.

See ["Configuring the platform launcher" on page 44](#) for details on available commands.

**Starting the Lotus Expeditor Client**

During installation of the Lotus Expeditor Client a new desktop icon titled Expeditor 6.1.1 or 6.1.2 will be created, which can be used to start the Lotus Expeditor Client.

Additionally, you can launch the Lotus Expeditor Client from the command line. By default, no arguments are required, but additional arguments can be provided. For a list of the supported command line arguments, see ["Configuring the platform launcher" on page 44](#). On Linux systems, if the Lotus Expeditor Client has been installed to the default directory, use the following command to launch the client:

```
/opt/ibm/lotus/Expeditor/rcp/rcplauncher
```

On Windows systems, if the Lotus Expeditor Client has been installed to the default directory, use the following command to launch the client:

```
C:\Program Files\ibm\Lotus\Expeditor\rcp\rcplauncher.exe
```

**Using the RCP daemon and Dcommands**

This section describes how the daemon works and how and existing process is joined when using a new personality.
When the Lotus Expeditor Client is running, you can execute commands from outside of the Expeditor user interface through the use of an Expeditor daemon. These commands are treated as additional arguments to the main rcplauncher executable. When an rcplauncher process is invoked with these additional commands and an Lotus Expeditor Client is already running, the rcplauncher process will use the daemon and inform the Lotus Expeditor Client to execute these commands instead of bringing up another full instance of the Lotus Expeditor Client and then invoking the commands within the new environment.

You can invoke additional personalities that will join the currently running Lotus Expeditor Client environment rather than creating another full instance. This is done using the daemon discussed in the previous section and is done to help conserve system resources.

As an example, let’s examine an office productivity suite of applications containing a word processor and a spreadsheet application. Each of these applications may be separate desktop icons, both of which use the rcplauncher executable but provide a different personality via the -personality parameter. Initially, the user needs to work on some spreadsheet data to generate the necessary data for a letter to be written so the user launches the spreadsheet. When nearing completion of the spreadsheet work, the user decides to begin work on the letter to ensure the layout will lend well to the data to be included, so the user now launches the word processing application. The rcplauncher process will recognize the Lotus Expeditor Client is already running with the spreadsheet personality and will not invoke another full instance. Instead, the rcplauncher process will use the daemon and have the currently running Lotus Expeditor Client open a new window, using the new word processing personality.

**Understanding workspace creation**

This section provides information about the creation and the contents of the IBM Lotus Expeditor workspace.

The workspace is used to store data in between platform launches, and also contains the configuration area for the platform. The configuration area stores information related to the applications and features installed in the platform.

The initial workspace is created during the initial install. The install program first installs a minimal set of features and plug-ins into the installation directory. These features are then used to launch the platform for the first time. During this first time launch, the initial workspace is created. The initial workspace and configuration are created based on the contents of the provisioning manifest. All of the features defined in the provisioning manifest are installed and enabled, and the resulting configuration information is stored in the workspace.

In multi-user installations, when additional workspaces are created following the initial workspace creation, the features installed during the initial workspace creation are used. With respect to the feature life cycle, this means that the features are not installed again but are only enabled. If you are providing your own features and they contain install handlers, they should perform most of the required work on the configure and unconfigure operations. For more information, see “Understanding feature life cycle” on page 145.

Two system properties defined in the .config/rcpinstall.properties file in the workspace provide specific information for the launcher to determine what happens when launching the platform.

**rcp.install.id**

Associates a version of the workspace with an installed version of the platform. Upon launch, the value of rcp.install.id is compared with the value of the rcp.install.id in the <install_dir>/rcp/rcplauncher.properties file. If these values are different, it is assumed that the workspace does not match the installed platform. See “Reusing an existing workspace” on page 24 for more information.
**provisioning.manifest.version**

Defines the version of the workspace configuration that is used. On launch, the value of the `provisioning.manifest.version` is compared with the value of the `provisioning.manifest.version` property in the `<install_dir>/rcp/rcplauncher.properties` file. If these values are different, then it is assumed that the provisioning manifest defined by the `rcplauncher.properties` file is more recent and needs to be applied to the workspace. The launcher then launches the platform using the `ProvisioningApplication` to perform the operations defined in the provisioning manifest. See “Using the Provisioning application” on page 239 for more information.

As the platform definition contains features associated with specific locales, only those features that apply to the current locale will be enabled. For example, if you launch the platform using the German locale `de`, the only features that will be enabled are those without a specific locale filter and those features that contain the filter specifying `de` as a specific locale. In a multi-user install scenario, you must first install all locale-related features prior to users creating a new workspace. See “Installing additional languages” on page 170 for more information.

**Reusing an existing workspace**

This section details how existing workspaces are handled.

Workspaces will be reused if the value of the `rcp.install.id` property in the `rcpinstall.properties` does not match the value of `rcp.install.id` property in the `rcplauncher.properties`.

This situation will generally occur if a workspace pre-exists on a system, and a new install of IBM Lotus Expeditor occurs.

If the launcher determines that the workspace does not match the installed version of the platform based on the `rcp.install.id` property, the launcher will take the following actions:

- The `.config/platform.xml` file will be copied to the workspace root and renamed to `platform.xml.old`.
- The `.config` directory will be removed.
- A new `rcpinstall.properties` will be created from the contents of the `rcpinstall.properties` supplied in the `com.ibm.rcp.base` plug-in.
- The `ProvisioningApplication` will be launched to create a new workspace:
  - The `ProvisioningApplication` will read the provisioning manifest defined by the `rcplauncher.properties` file and will attempt to create a new workspace using the features already installed on the file system, just as if the workspace had not previously existed.
  - The `ProvisioningApplication` will read the `platform.xml.old` file that was created and will attempt to enable each of the features included in the shared site and each of the features included in the user applications site.
  - The `ProvisioningApplication` will exit and the launcher will restart the installed platform.

**Launching the Lotus Expeditor Client from a network drive**

This topic describes how to start the Lotus Expeditor Client from a network drive.

When the Lotus Expeditor Client has been installed onto a shared network drive, no icons are created or available for users to do the initial launch of the Lotus Expeditor Client. The initial launch must be performed by locating the `rcplauncher.exe` (Windows) or `rcplauncher` (Linux) program, and starting the program.

For example, on Windows, if you have a network drive `X:` that contains the directory `X:\Expeditor`, switch to the `X:\Expeditor\rcp` directory, and start the `rcplauncher.exe` program. The workspace will be created in the normal location. As part of the initial creation of the workspace, icons will be added to the desktop to enable subsequent launches to occur using the shortcuts.
In the event that the drive letter that refers to the network drive changes after the initial launch, the created icons will no longer allow you to launch the Lotus Expeditor Client. If the drive letter is permanently changed, you can update the shortcut to change the launch location.

**Launching the Lotus Expeditor Client using a non-standard workspace location**

The default workspace location is computed based upon the operating system and the defined user.

If your workspace needs to be located at an alternative location, then start the launcher by specifying the "-data" parameter. For example, if you want to store a workspace on a USB drive connected to the system on drive G:, you can start the Lotus Expeditor Client by specifying `replauncher -data G:\workspace`. As this is a non-standard location for the workspace, you must either update the shortcut used to launch the Lotus Expeditor Client, or always use a console or Run menu to launch the Lotus Expeditor Client.

The same workspace can be referenced from different locations. For example, if the workspace is created on a USB-drive, it may attach as drive G: on one machine, and as D: on another. You will need to launch the Lotus Expeditor Client and specifically identify the data directory on each launch.

See "Overriding the workspace directory location" on page 20 for more information.

**Using links to composite applications**

Use the CAI protocol to launch IBM Lotus Expeditor.

Composite Applications are URL-addressable with URLs beginning with `CAI://`. This is known as the Composite Application Infrastructure (CAI) protocol. For the browser to understand the CAI protocol, CAI must be registered with the operating system. In Windows, the registry file `cai_register.reg` is part of the IBM Lotus Expeditor installation. IBM Lotus Expeditor contains information to be written to the Windows registry. In Linux, a similar technology, called GConf, is used to register CAI with the operating system. On a multi-user installation, the first time the Lotus Expeditor Client is run by each user, this registration (enablement) is executed.

Thereafter, when a URL beginning with `CAI://` is used in the browser, IBM Lotus Expeditor will open (if not already open) and load the specified composite application. See "Configuring application layout" on page 143 for more information. In addition, links starting with `CAI://` that are included inside document types which support links on the operating system will also be enabled, resulting in IBM Lotus Expeditor opening and loading the specified composite application.

Within IBM Lotus Expeditor, use the Edit → Copy As menu items (enabled when a composite application is opened) to create these `CAI://` links.

**Installing the Lotus Expeditor Client on devices**

This section explains how you, as administrator, can assist users in installing Lotus Expeditor Client on supported devices.

Note that devices have different capabilities for running large application environments like Lotus Expeditor Client. You may want to pre-approve certain devices for users based on the application load you expect them to run.

**Installing the core runtime**

There are several ways to install the Lotus Expeditor Client core runtime on a supported device:
Installing from a CD
For Windows Mobile and WinCE devices, you can provide a CD to end-users. The user must have a PC with ActiveSync installed and their device connected. The user runs a setup program on the PC which transfers a setup program to the device where installation continues. This method is only appropriate when there is a one-to-one relationship between the PC and device. Alternately, you could provide users with one of the following operating system-based Lotus Expeditor Client setup files using some other mechanism.

<table>
<thead>
<tr>
<th>Setup file</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPD-wm-setup.exe</td>
<td>For Windows Mobile devices</td>
</tr>
<tr>
<td>Expeditor-wince5-setup.exe</td>
<td>For WinCE devices</td>
</tr>
</tbody>
</table>

Installing from a storage card
For Windows devices, you can provide users with a storage card containing one of the following operating system-based Lotus Expeditor Client cab files. Users can insert the storage card into their device and start the installer using File Explorer.

<table>
<thead>
<tr>
<th>Setup file</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPD-Core-wm2003.cab</td>
<td>For Windows Mobile 2003 devices</td>
</tr>
<tr>
<td>XPD-Core-wm.cab</td>
<td>For Windows Mobile 5 devices</td>
</tr>
<tr>
<td>Expeditor-Core-wince5.cab</td>
<td>For WinCE devices</td>
</tr>
</tbody>
</table>

For Nokia E90 devices, you can provide a storage card containing Nokia eRCP, the IBM Certificate Installer, and the IBM Lotus Expeditor Core S60 update site. See the Lotus Expeditor Client Release Notes for the Web addresses of Nokia eRCP and IBM Certificate Installer. The IBM Lotus Expeditor Core S60 update site must be copied from the IBM Lotus Expeditor CD. Copy the s60 folder from \device\install to \downloads on the storage card. Users can insert the storage card into their devices and open Nokia eRCP to install it. Then open the IBM Certificate Installer to install it. For more information on installing Nokia eRCP, see the Nokia eRCP Installation and Configuration Guide included with the Nokia eRCP package. Users launch eRCP Application Manager to install the IBM Lotus Expeditor feature from the S60 update site. For more information on the Nokia eRCP Application Manager, see the Nokia eRCP User Guide included with the Nokia eRCP package.

**Note:** The support for the Nokia E90 is provided as early release code for internal evaluation and testing. The support for the Nokia E90 may not be used for production purposes.

Installing from a Web site
For Windows devices, you can set up a Web site that provides one of the following IBM Lotus Expeditor cab files. Users use Internet Explorer on the devices to browse to your Web site and download the appropriate setup file based on the device operating system. The setup file automatically runs and installation starts on the device.

<table>
<thead>
<tr>
<th>Setup file</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expeditor-Core-wince5.cab</td>
<td>For Windows CE 5 devices</td>
</tr>
<tr>
<td>XPD-Core-wm2003.cab</td>
<td>For Windows Mobile 2003 devices</td>
</tr>
<tr>
<td>XPD-Core-wm.cab</td>
<td>For Windows Mobile 5 devices</td>
</tr>
</tbody>
</table>

**Note:** For information about setting up your Web server, see "Installing with the Network Client Installer" on page 29.
For Nokia devices, you can set up a Web site that provides the IBM Lotus Expeditor Core S60 update site. Users use the Nokia Web browser to download and install Nokia eRCP and the IBM Certificate Installer. Then, launch eRCP Application Manager to install Expeditor from the S60 update site.

If you are migrating server-managed devices from IBM Lotus Expeditor 6.1 or 6.1.1 to 6.1.2, after the migration, you should schedule an inventory job so that the management server will obtain updated information about the managed devices.

**Installing Lotus Expeditor Client applications**

There are several ways to install applications that run on the Lotus Expeditor Client onto devices. The method you chose depends on how your Clients will be managed. In all cases, it is your responsibility to provide the dependencies that an application needs to execute. These may include optional features from the Lotus Expeditor Client for Devices update site.

**Enterprise Managed Clients**
You remotely manage devices. In this case, you use the Device Manager server console to package applications and create jobs that are automatically deployed to devices. See "Managing using the Device Manager server" on page 181 for more information. When an application is installed on a device, the user might see a pop-up message informing him to restart Lotus Expeditor Client (or close Nokia eRCP for Nokia devices). If a restart is required, the new application will be launchable after the restart. Otherwise, it is launchable immediately.

**Remote Update Site**
You can package applications in an update site that you make available on a Web server. See "Creating an update site" on page 151 for more information. The user uses Application Manager (or eRCP Application Manager on Nokia devices) to browse the remote update site and install applications from the site. Note that you must tell the user the specific Web address to browse.

**Local Update Site**
You can package applications in an update site that you make available on a storage card. The user inserts the storage card into the device and uses Application Manager (or eRCP Application Manager on Nokia devices) to open the site and install applications. Note that for Windows devices, it is best to place update sites in a folder in the root of the storage card. The folder name can thus identify the update site and you can place more than one site on a card if needed. Sites should not be placed more than one folder deep. For Nokia devices, the update site must be in a folder placed in the \downloads directory.

For Windows Mobile devices, you can also place an update site in the My Documents folder on the device.

For WinCE devices, you can also place an update site in any location on the device.

**Note:** When installing, updating, or uninstalling features, the user may be advised to restart the client runtime. This is generally not necessary when installing new features. However, updates and uninstalls will not be effective until a restart occurs. If a user has a batch of updates or uninstalls to perform, it is acceptable and more efficient to delay restarting until the last operation has been completed.

**Client prerequisites**
This section describes the client prerequisites for installing Lotus Expeditor Client on a device.

You can run the Lotus Expeditor Client on devices with the following Windows operating systems:

- Windows Mobile 5 (Pocket PC and Phone Edition)
- WinCE 5.0 Professional
Specific units tested for this release:
- Dell Axim (X50v and X51v)
- HP iPAQ (2790 and 4700)
- I-Mate JAsJar
- Unitech PA962
- Symbol MC3090R-LC48S00GER
- Symbol MC9090-GF0HBEGAA2WW

Specific units supported: Nokia E90

Note: The support for the Nokia E90 is provided as early release code for internal evaluation and testing. The support for the Nokia E90 may not be used for production purposes.

You can install the IBM Lotus Expeditor to your device’s file system or to a storage card. You must have a minimum of 12MB of file space available. An additional 14MB of file system space is temporarily required to install from a cab file. A minimum of 12MB of available physical memory is required. However, the minimum requirements will likely only let you run smaller Rich GUI applications. It is recommended that your device have 16MB of file system space and 16MB of physical memory available.

You can run the Lotus Expeditor Client on Nokia E90 devices.

To install the Lotus Expeditor Client on an E90, you must first install Nokia eRCP and the IBM Certificate Installer. See the Lotus Expeditor Client Release Notes. Lotus Expeditor Client for the Web addresses of these components. After Nokia eRCP is installed, you must have a minimum of MB of file system space and 2 MB of physical memory available. However, the minimum requirements will likely only let you run smaller Rich GUI applications. It is recommended that your device have 8 MB of file system space and 8 MB of physical memory available.

Uninstalling
You can uninstall Lotus Expeditor Client from a device.

For Windows Mobile devices, if you installed using a setup.exe file, connect your device using ActiveSync and uninstall by running the setup.exe again, choosing to remove the Lotus Expeditor Client. If you installed from a storage card or Web site, then on your device select Start → Settings → System → Remove Programs. Select IBM Lotus Expeditor Client and tap Remove. Do not remove the \workspace directory if you want to retain application information and device ID.

For WinCE devices, if you installed using a setup.exe file, connect your device using ActiveSync and uninstall by running the setup.exe again, choosing to remove the Lotus Expeditor Client. If you installed from a storage card or Web site, then on your device select Start → Settings → Control Panel → Remove Programs. Select IBM Lotus Expeditor Client from the list of applications. Do not remove the \workspace directory if you want to retain application information and device ID.

For Nokia E90 devices, launch the eRCP Application Manager. Select Manage Features. Select the Lotus Expeditor Client features you want to remove. Select Options → Uninstall Feature.

Note: The support for the Nokia E90 is provided as early release code for internal evaluation and testing. The support for the Nokia E90 may not be used for productive purposes.

Launching the Lotus Expeditor Client on devices
Review how to launch the Lotus Expeditor Client on Windows and Nokia E90 devices.
You do not need to directly launch the Lotus Expeditor Client on devices. Instead the Lotus Expeditor Client environment is started automatically when Lotus Expeditor Client applications are started. Note that the device cannot be remotely managed until an Lotus Expeditor Client application is started.

On Windows devices, the Lotus Expeditor Client remains running while any IBM Lotus Expeditor application is still open.

On Nokia E90 devices, once Nokia eRCP is started, the Lotus Expeditor Client remains running until the device is powered off.

**Note:** The support for the Nokia E90 is provided as early release code for internal evaluation and testing. The support for the Nokia E90 may not be used for production purposes.

---

### Installing with the Network Client Installer

Install the Network Client Installer so that users can remotely install the Lotus Expeditor Client, install the administrative components needed on your Portal server, and manage the Lotus Expeditor Client.

**Note:** **Running command-line tools on Windows when configured for Hungarian:** By default, when Windows is installed with Hungarian, the code page for the command prompt does not contain all the characters needed to display the Hungarian translations provided by the ArchiveCreator and SiteXMLUpdater tools. To use these tools, configure the command prompt by following these steps:

1. Set the font to Lucinda Console.
2. Set the code page to 1251 by running the command `chcp 1251`.

The text for the command-line tools provided by the Network Client Installer will now display correctly for Hungarian.

### Network Client Installer client prerequisites

Ensure that each machine on which you plan to install the Network Client Installer meets these requirements.

Minimum hardware requirements include the following for the client machines:

- 10 MB free disk space (Portal only)
- 200 MB free disk space (Web only)
- 210 MB (Web and Portal)
- Requirements for Web: IBM HTTP server or another supported HTTP server
- Requirements for Portal: Portal 6.0

Windows IA32 and Linux IA32 environments are supported as follows:

- Microsoft Windows 2000 Professional Service Pack 4 (for the Lotus Expeditor Client desktop runtime only)
- Microsoft Windows XP Professional Service Pack 1 and 2
- Microsoft Windows XP Home Edition Service Pack 1 and 2
- Microsoft Windows XP Tablet PC Edition 2005
- Red Hat EL 4.0 WS with GTK support – Update 4
- Red Hat EL 5.0 WS with GTK support
- SUSE Linux Enterprise Desktop 10
- Microsoft Windows Vista
Installing the Network Client Installer

You can install the Network Client Installer on Lotus Expeditor Server, a standalone Web server, or a Portal server.

Before you begin you should understand the configuration of these servers to know which choices are best for you. You must also have an active account on the server.

To install the Network Client Installer, follow these steps:

1. Insert the Lotus Expeditor Client CD into the CD-ROM drive.

   \textbf{Note:} For Portal clustered environments, install the Network Client Installer on only one Portal node in the cluster.

2. If autorun is enabled on your system, a browser opens and displays a welcome page. If autorun is disabled, launch the file that is appropriate for your server operating system. The Network Client Installer is located on Lotus Expeditor Client CD in the \textbackslash{}nci directory:

   \begin{description}
   \item[Windows] Launch \textbackslash{}nci\install\setupwin32.exe
   \item[Linux] Launch \textbackslash{}nci\install\setuplinux.sh
   \item[AIX®] Launch \textbackslash{}nci\install\setupaix
   \item[Linux on System p™] Launch \textbackslash{}nci\install\setupLinuxPPC.sh
   \end{description}

3. Follow the prompts to install the Network Client Installer.

   \textbf{Note:} Do not install the Network Client Installer in the multiple locales NL path. Valid characters for an installation directory are a-z, A-Z, 0-9, ‘-’ (dash), ‘_’ (underscore), and space (Windows only). Double-byte characters are not valid.

   If you choose to install on the Web server, you are prompted for the HTTP server document root location, the host name, and the URL to access the document root. You also have the choice to set the Device Management agent URL, as well as the Portal URL.

   If you choose to install on the Portal server, you are prompted for the Portal server home directory and the administrative user ID and password for the Portal server. Specify these settings as they pertain to your environment.

4. \textbf{6.1.2} For \textit{Portal clustered environments only}, complete the following steps:
   \begin{itemize}
   \item[a.] Wait for the Deployment Manager to synchronize the Network Client Installer portlets to the other nodes in the cluster, or force synchronization.
   \item[b.] Activate the deployed portlets:
   \begin{description}
   \item[UNIX] Run the following command from the \texttt{portal_server_root/config} directory:
     \texttt{.WPSconfig.sh activate-portlets}
   \item[Windows] Run the following command from the \texttt{portal_server_root/config} directory:
     \texttt{WPSconfig.bat activate-portlets}
   \end{description}
   \end{itemize}

The Network Client Installer is installed on the server and the server is ready for a client to install.

\textbf{If you are installing on the Web server}, the following Network Client Installer components are deployed on the server:
\begin{itemize}
\item Lotus Expeditor Client update site
\item Lotus Expeditor Client installers
\item Download applet
\end{itemize}
• The configuration files for the download applet that contain the host URL are updated with information gathered during the installation.

If you are installing on the Portal server, the following Network Client Installer components are deployed on the Portal server:
• Portlets for Portal-administered client
• Download portlet for client install
• Application template

After the Network Client Installer is installed, the file layout is as follows:
/Expeditor
Contains the program root folder files.
/Expeditor/applet
Contains files for the download applet.
/Expeditor/install
Contains client installers and option files.
/Expeditor/updates
Contains the Eclipse update site.

Related information
"Notifying users on how to install from the Web using Windows and Linux" on page 33
As administrator, you need to inform your users on how to install the Lotus Expeditor Client from the Web using Windows and Linux.

"Notifying users on how to install from the Web using supported devices" on page 34
As administrator, you need to inform your users on how to install the Lotus Expeditor Client from the Web using supported devices.

Installing the Network Client Installer when using SSL with a self-signed test certificate
If you have configured your Web server or Portal server to use Secure Sockets Layer (SSL) and are using a self-signed test certificate, you must create a truststore and define it to the Network Client Installer before running the Network Client Installer to ensure that SSL communications will occur properly between the Network Client Installer install program and the Web server or Portal server.

If the Portal environment is HTTPS(10035) and 10038 port is disabled, before installing the Network Client Installer, you must validate that the entries in the <WP_HOME>/config/wpconfig.properties file are correct. For example:
XmlAccessPort=10035
xmlAccessProtocol=https

Steps 1-5 below are specifically for IBM HTTP Server (IHS). If you are using a different HTTP server, consult that documentation for how to create a truststore, and execute Step 6 below.
1. Extract the self-signed certificate from the IBM HTTP Server (IHS) if you are installing onto the Web server.

Launch ikeyman from <ihs_home>\bin and open the keystore defined in the httpd.conf file under the Keyfile directive. For example, the keyfile C:/Program Files/IBM HTTP Server/selfCert/serverkey.kdb should look similar to the following:

# Extract the public self-signed certificate key.
# Click Personal Certificates in the menu and select the self signed certificate you are using.
# Click Extract Certificate. Extract the certificate to a file:

Data type
Base64-encoded ASCII data
Certificate file name
cert.arm
2. Extract the self-signed certificate from Portal.
   Launch ikeyman from <ihs_home>/bin and open the keystore defined in for Portal. The keyfile
   C:\ibm\WebSphere\profiles\wp_profile\etc\DummyServerKeyFile.jks should look similar to the
   following:
   # Extract the public self-signed certificate key.
   # Click Personal Certificates in the menu and select the self signed certificate
   # you are using
   # Click Extract Certificate. Extract the certificate to a file:
   
   Data type
   Base64-encoded ASCII data
   Certificate file name
   cert.arm
   Location
   C:\ibm\WebSphere\profiles\wp_profile\etc
   
   # Click OK.

3. Create a new keystore.
   Launch ikeyman from <ihs_home>/bin
   # Click Key Database File > New from the iKeyman menu.
   # Enter the following settings:
   
   Key database type
   JKS
   File name
   key.jks
   Location
   c:\testKeyStore
   
   # Click OK.

4. Import the certificate from the IBM HTTP Server.
   # Click Signer Certificates from the list and click Add. This action imports the
   # public certificate previously extracted from the HTTP server keystore file.
   
   Data type
   Base64-encoded ASCII data
   Certificate file name
   cert.arm
   Location
   c:\program files\IBM Http Server\selfcert
   
   # Click OK. You are prompted for a label name that represents the trusted signer public certificate.
   # Enter a label for the certificate: IHS Self Signed certificate

5. Extract the self-signed certificate from Portal
   Launch ikeyman from c:\Program Files\IBM Http Server\bin
   Open the keystore defined in for portal
   Keyfile C:\ibm\WebSphere\profiles\wp_profile\etc\DummyServerKeyFile.jks
   
   # Extract the public self-signed certificate key.
   # Click Personal Certificates in the menu and select the WASplugin certificate that you just created.
   # Click Extract Certificate. Extract the certificate to a file:
   
   Data type
   Base64-encoded ASCII data
   Certificate file name
   cert.arm
   Location
   C:\ibm\WebSphere\profiles\wp_profile\etc
   
   # Click OK.

6. Create an appropriate SP file.
Using an editor, create a text file with the same name as the Network Client Installer install launcher, but with the extension .sp:

**Windows:**
- setupwin32.sp

**Linux:**
- setuplinux.sp

**AIX:**
- setupaix.sp

**Linux on System p:**
- setupLinuxPPC.sp

Add these lines:

```java
javax.net.ssl.trustStore=C:\testKeyStore\key.jks
javax.net.ssl.trustStorePassword=your keystore password
javax.net.ssl.trustStoreType=jks
```

When you launch the installer, it will now communicate correctly using SSL with the self-signed certificates.

Since the following command is necessary to successfully install the Network Client Installer in a Portal SSL environment where port 10038 is disabled, then that information should be communicated to your users.

```
./setupLinuxPPC.sh -is:javaconsole -V NCI_PORTAL_HOME_URL_PORT=10035 -V NCI_PORTAL_HOME_URL_PROTOCOL=https
```

### Notifying users on how to install from the Web using Windows and Linux

As administrator, you need to inform your users on how to install the Lotus Expeditor Client from the Web using Windows and Linux.

Your users should be informed of the following steps:

1. Provide users with the URL of your customized installation Web site. Use the sample HTML template provided and customize it for your environment. The default is http://<server name>/Expeditor/applet/index.html.

2. If their browsers do not support Java, they will be prompted to save a zip file that contains the installer and its associated files. Instruct them to unzip this file and execute the appropriate installer: install.bat (for Windows) or install.sh (for Linux). If you have updated the download.properties file to pass parameters to the installer, you should give these instructions to your users. The command line options will not be contained in the zip file.

3. Upon completion, the client installer contacts the server and completes the installation. If the update site is in a protected area, the client will prompt for user ID and password.

### Special considerations for Windows Vista

Installing the Lotus Expeditor Client from the Web using Windows Vista in protected mode is not supported. Follow these steps as a work-around:

1. Place your Web server in a trusted zone using the Internet Explorer configuration. This will disable protected mode for this site and allow the installation to succeed.

2. Direct your users to access the following site:


   where <server> is the server name and <language> is the language.
Notifying users on how to install from the Web using supported devices

As administrator, you need to inform your users on how to install the Lotus Expeditor Client from the Web using supported devices.

Your users should be informed of the following steps:

1. Provide users with the URL to your customized installation Web site. Use the sample HTML template provided and customize it for your environment. The default is http://<server name>/Expeditor/applet/DownloadListIndex.html.

2. Select the appropriate link for your platform from the list at the bottom. This causes a .cab file to be downloaded to the device. When the device asks if the file should be automatically installed, users should select **yes, install the file automatically**. After the installation is complete, users must manually configure the rest of the settings.

3. Upon completion, the users of enterprise managed devices must set Enterprise Management preferences. This can be done by installing a configuration file you provide to them or by setting the fields in the Enterprise Management preference page. This preference page is accessible on the device from the Application Manager by selecting **Command → Preferences**. See ["Configuring the Enterprise Management Agent" on page 18](#) for more information on Enterprise Management Agent parameters.

Silently installing the Network Client Installer

You can set up the Network Client Installer installation program to run silently if you do not want to interact with the installation wizard.

To do this, you should install using the installation wizard, answering all the questions as how you would like the silent install to run. After the installation is complete, copy the installNCI.rsp file from the <destination>/package directory, where <destination> is the directory in which you chose to install the Network Client Installer.

**Note:** Installer executables are located in the nci/install/ directory of the CD.

- On a Windows system, run the setupwin32.exe file.
- On a Linux system, run the setuplinux.sh file.
- On an AIX system, run the setupaix file.
- On a Linux system for System p, run the setupLinuxPPC.sh file.

Choose from two ways to install silently:

- **Important:** If you choose this option, your Portal password will remain in clear text in the response file after the installation has completed. This is not the recommended approach.

1. Using a text editor, open the response file that you copied earlier, and <destination_dir>/package/installNCI.rsp, replace <enter your password> with your password in the NCI_PORTAL_PASSWORD property, and save the file.

   For example, NCI_PORTAL_PASSWORD=expeditor.

2. Run the following command:

   `<setup executable> -silent -options installNCI.rsp`

   For example, on Linux:
   `setuplinux.bin -silent -options installNCI.rsp`

   - Use a command line option.

   1. Run the following command:

      `<setup executable> -silent -options installNCI.rsp -V NCI_PORTAL_PASSWORD=<portal password>`

      For example, on AIX:
setupax -silent -options installNCI.rsp -V NCI_PORTAL_PASSWORD=myPortalPassword

Note: Because this is a silent install, no command line or graphical output is created. To view the progress of the install or to see any error messages generated, view the install log file, which is located in the temp folder defined by your operating system. In this temporary directory, the file NCI/logs/masterInstallLog.txt will contain the output from the installation.

Uninstalling the Network Client Installer

This topic describes how to uninstall the Network Client Installer. Before you uninstall the Network Client Installer, check that the installation completed successfully.

To uninstall the Network Client Installer, follow these steps:

1. Locate and launch the Network Client Uninstaller program. Depending on how the product was installed, the uninstaller file is located in either the _uninst2 or _uninst3 directory in your installation path (specified by \_uninstn as shown).

   Windows
   "install_path\_uninstn\uninstaller.exe.

   Linux  "install_path/_uninstn/uninstaller.sh

   AIX  "install_path/_uninstn/uninstaller.bin

   Linux for System p
   "install_path/_uninstn/uninstaller.sh

Note: To uninstall the Network Client Installer on WebSphere Portal with SSL enabled, follow these steps:
   a. Copy the .sp file generated during the Network Client Installer installation to the
      nci_home/_uninstn directory that contains the uninstaller program, where nci_home is the
      Network Client Installer installation directory and n is a number.
      b. Rename the .sp file to un installer.sp so that it matches the launcher’s base file name.
      c. Launch the uninstaller program and complete all required steps.

2. Follow the prompts.

The Network Client Installer is uninstalled.

Silently uninstalling the Network Client Installer

You can silently uninstall the Network Client Installer installation program.

If you need to silently uninstall the Network Client Installer, you can should use the appropriate uninstall launcher from above, together with the silent install response file referenced in "Silently installing the Network Client Installer" on page 34.

For example, on Windows:
C:\program files\IBM\Lotus\Expeditor\NCI\uninst3\uninstaller.exe -silent -options installNCI.rsp -V NCI_PORTAL_PASSWORD=expeditor

Upgrading the Network Client Installer

Consider these things first before upgrading.

If you had a previous version of the Network Client Installer (NCI) installed on your system, and you are installing this version as an upgrade, there are some things you should consider:

• If a previous version of NCI is installed, it will be detected. The new version will install directly
  without the need to manually uninstall the previous version.
• The client update site that is installed on the web server will be overwritten with the update site on the client CD. This means that if you have added features to this site and use the siteXmlUpdater tool, you will need to re-run this tool after the update.

• The client installers and manifest files will be overwritten with the client installers from the client CD. This means that if you have customized the provisioning manifest files in the install/deploy directory, you will need to re-customize these files after the update.

Updating the installation information on the Web server

If you move your Web server or change other system information, you will need to update the configuration on the Web server.

To update the installation information:
1. Update the installer.root property in the download.properties file.
   If you want to change the way the client installer is executed, you can update the win32.downloads.execute and the linux.downloads.execute property. This is useful for passing an options file or other parameters to the installer. See “Using options files” on page 14 for more information.

2. Run the ArchiveCreator tool located in the <Expeditor_install_root>/bin directory.

The ArchiveCreator tool recreates the zip files that are downloaded when the Lotus Expeditor Client cannot run the download applet.

Installing the Lotus Expeditor Client for Windows and Linux on Portal

You can install the Lotus Expeditor Client for Windows and Linux on Portal.

Before you install the Lotus Expeditor Client, review system requirements and verify that you have the correct version of Portal Server installed and configured. Check that the Network Client Installer software does not previously exist on the server. A supported HTTP server must also be installed. It can be on another computer in the network.

You must also have an active account on the server.
1. Install the Network Client Installer from the Lotus Expeditor Client CD.

2. Select Web content and Portal content from the list of features.
   
   Note: If you are installing in a clustered environment, select only Portal content from the list of features. Web content must be installed on the remote Web server.

3. Use the Portal Administration portlets to place the downloads portlet on a page that is accessible by your users.

4. In a clustered environment, edit the value for wed.download.url of the Download portlet to the URL of the remote Web server installation site. For example, http://<server name>/Expeditor/applet/index.html.

The Lotus Expeditor Client is ready to be installed on Portal.

Notifying users on how to install the desktop client from Portal

As administrator, you need to notify your users on how to install from Portal.

Your users should be informed of the following steps to install the Lotus Expeditor Client from Portal.
1. Log in to Portal using a browser, and access the Downloads portlet.
2. Select the install link.
3. If their browsers do not support Java, they will be prompted to save and execute the installer.
4. Users may be prompted to restart the Lotus Expeditor Client.
**Applying updates**

You can apply updates to the Network Client Installer from the server.

Before you install the Lotus Expeditor Client, check that either the Lotus Expeditor Server or Portal Server 6.0 or greater is installed and configured. Check that the Network Client Installer software has been deployed on the Lotus Expeditor Server.

1. Download and install the Update installer if it is not already present on the server.
2. Download the Network Client Installer update package.
3. Run the Update installer and select the update to the Network Client Installer.

**Applying fixpacks and individual fixes**

You can apply fixpacks and individual fixes to update Network Client Installer bundles.

Before you install the update to the Lotus Expeditor Client, ensure that the Network Client Installer has been deployed to your Web server.

1. Download the fixpack.
2. Unzip the IFix on top of the existing Eclipse update site (located under `<docroot>/expeditor/updates/platform`).
3. Run the siteXMLUpdater tool located in the `<nci_install_dir>/bin` directory. This updates the `site.xml`.

**Using options files**

If you want to change the options of the client install that results from downloading from the Web Server, you can edit the options files located in the `<docroot>/Expeditor/install` directory on the Web server.

See ["Using options files" on page 14](#) for the format of these options files.

To choose the command-line options for the client install from the server, edit the `download.properties` file under the `<docroot>/Expeditor/applet` directory.

After you edit this file or the options files, run the Archive Creator tool under `<nci_install>/bin` to regenerate the archives for non-java enabled browsers.

---

**Installing and launching the Expeditor Client in a Citrix environment**

The Lotus Expeditor desktop client can be installed on a Citrix Presentation Server 4.5 running on Windows Server 2003 and, subsequently, published as an application to allow Citrix users access. This section describes the procedures involved for such an installation.

**Prerequisites**

This topic lists prerequisites and recommendations to install the Lotus Expeditor desktop client on a Citrix Presentation Server 4.5 successfully.

Prerequisites are as follows:

- Windows Server 2003 with Citrix Presentation Server 4.5
  - **Attention:** For a new Citrix installation, it is highly recommended to test the server by publishing an application, such as Adobe® Acrobat® Reader, before attempting to publish Lotus Expeditor.
- Citrix Presentation Server 4.5

Notes:

- Lotus Expeditor 6.1.2 only supports the Web/ICA deployment technology.
- Lotus Expeditor 6.1.2 only supports being published as an application.
- Lotus Expeditor 6.1.2 does not support multiple client monitors with Citrix Presentation Server 4.5.
- If a farm of Citrix servers is being used, it is recommended to also employ Windows Roaming Profiles to ensure that the user has access to their Lotus Expeditor-specific information, regardless of which server the Expeditor client is running on in the farm.

### Installing Expeditor Client on Windows Server 2003 / Citrix Server
This topic describes how to install the Expeditor Client in a Windows Server 2003 / Citrix Presentation Server 4.5 environment.

The initial installation steps are similar to any other multi-user Lotus Expeditor installation:
- Member of Administrators group perform the installation
- Multi-user
- Configure Home Portal, Enterprise Management, and Single Sign-on (SSO) if desired
- Launch Lotus Expeditor after installation to ensure functioning properly
- Install all additional Lotus Expeditor-based applications, which can be accessed by users
- Exit Lotus Expeditor

### Configuring Citrix Presentation Server to publish Lotus Expeditor as an application
This topic describes how to use the Citrix Presentation Server wizard to publish Lotus Expeditor as an application.

Citrix Presentation Server 4.5 provides a wizard for publishing applications. You can launch the wizard from within the Citrix Access Management Console. Recommended settings for each wizard panel are as follows:

**Note:** Settings assume the default installation location: C:\Program Files\IBM\Lotus\Expeditor\  
- **Welcome:** There are no configuration options. Continue to the next panel.  
- **Name:** Type a display name, such as **Lotus Expeditor** and complete the application description.  
- **Type:** In the Application section, select **Accessed from server** and **Installed application.**  
- **Location:** Complete the following fields:  
  - Command line: C:\Program Files\IBM\Lotus\Expeditor\rcp\rcplauncher.exe  
  - Working directory: C:\Program Files\IBM\Lotus\Expeditor\rcp  
  - Isolate application: Disable this check box.  
- **Servers:** Add one or more configured Citrix server names to deliver Lotus Expeditor to client machines.  
- **Users:** Add one or more user names, which require access to Lotus Expeditor.  
- **Shortcut presentation:**  
  - **Application icon:**  
    - The rcplauncher.exe binary does not contain icon information to allow branding and customization. Therefore, it is necessary to point to a specific Windows ICO file.  
    - The default Lotus Expeditor branding icon is located at C:\Program Files\IBM\Lotus\Expeditor\rcp\eclipse\features\com.ibm.rcp.personality.default.branding.feature_version\win32\xpd.ico  
  - **Application shortcut placement:** You can add the previously specified icon to the client machine’s Start Menu or as a Desktop shortcut, but is not necessary. If the icon is not added to either location, users must access the Citrix Web Interface to log in and, subsequently, launch Lotus Expeditor.  
- **Publish immediately:** Click **Finish.** Lotus Expeditor is available for Citrix users to access.
Accessing Lotus Expeditor remotely through the Citrix Web interface

This topic describes how Citrix clients can access Lotus Expeditor remotely.

After completing the installation and application publication steps, Citrix clients can access Lotus Expeditor by completing the following steps:
1. Open Internet Explorer and load the Citrix server Web address.
2. Install the Citrix ICA Client package, if it is not installed already. This might require a browser and/or system restart depending on the installation options selected.
3. Log in to the Citrix server. Lotus Expeditor is displayed in the Applications panel.
4. Click the Lotus Expeditor icon to launch the Citrix ICA Client. The Lotus Expeditor platform is launched.

Note: If different Display name or Application shortcut placement options were selected during configuration, the user might need to look for a different icon, or perform additional navigation operations on the Citrix Web interface, before the Lotus Expeditor platform is launched.

Citrix environment limitations

This topic lists limitations when running the Lotus Expeditor desktop client in a Citrix environment.

The Lotus Expeditor desktop client in a Citrix environment functions similar to any other multi-user installation. However, the following limitations are introduced, primarily due to multiple users concurrently accessing the platform:
- Lotus Expeditor 6.1.2 does not support the Lotus Expeditor Restricted Workbench functionality in a Citrix environment.
- The installation directory (for example, C:\Program Files\IBM\Lotus\Expeditor) must be properly secured to ensure that only members of the Administrators group can add, modify, and delete files.
- To prevent platform corruption, only one member of the Administrators group should manage/provision the Lotus Expeditor platform installation at any given time.

Changing the virtual machine

The default virtual machine used by the IBM Lotus Expeditor platform is the jclDesktop runtime environment. This virtual machine provides sufficient capabilities for the default install set of IBM Lotus Expeditor components. If you need more function, such as Swing, or AWT programming libraries that are provided by the Java 2 SE 5.0 virtual machine (VM), follow the instructions in this section to install the IBM Device Runtime Environment (DRE), a separately available product.

The IBM Device Runtime Environment product contains two features that are applicable to the Lotus Expeditor Client platform. The first is the J2SE Core JVM Feature, which provides the Java 2 SE 5.0 VM capabilities. The second feature is the JavaServer Faces Feature, an alternative to the Apache MyFaces support.

To use either of these components, you must install them into the Lotus Expeditor Client.

Installing DRE components into the Lotus Expeditor Client

Install the DRE using either the Install/Update UI or using the provisioning manifest. The Install/Update UI is useful for installing a DRE into only one runtime, but for automation purposes, you should consider using the provisioning manifest.

To install DRE components using the Update Manager UI:
1. Start the Lotus Expeditor Client. Then select File → Application → Install
2. From the Install/Update (Feature Updates) dialog, select Search for new features to install. From the Install (Update Sites to Visit) dialog, create a new Update site definition. Select a New Remote Site if the DRE has been made available using a Web site. Otherwise, select a New Local Site. Browse to the location of the DRE (CD-ROM drive letter or location where it might have been unzipped on a mapped drive letter or mount), then select updates/client. Select Finish. When the Install (Search Results) dialog appears, select the Core JVM Feature – J2SE feature, then select Next. (You can also select the JavaServer Faces (JSF) feature at this point). If the license terms are agreeable, select I accept the terms in the license agreement, then select Next. Then select Finish to complete the installation.

A dialog informing you that the feature is not signed might appear. Select Install on this dialog to continue.

3. To perform a more automated installation of the DRE, you can either customize the provisioning manifest for install [Customizing basic installation” on page 18] or use the ProvisioningApplication to install the DRE using a provisioning manifest [Using the provisioning manifest” on page 160].

For example, you could use the following provisioning manifest contents in file C:\core.jvm.feature-j2se.xml:

```xml
<install>
  <installfeature id="core.jvm.feature-j2se" required="true">
    <requirement>
      <feature action="install"
        id="com.ibm.rcp.jvm.feature"
        match="greaterOrEqual" url="file:\x:updates/client/site.xml"
        shared="false"
        version="2.8.0.0-200704260508"/>
    </requirement>
  </installfeature>
</install>
</object-data>
</domain-object>
</ibm-portal-composite>
```

where \( \text{x} \): is the CD-ROM drive containing the DRE CD-ROM.

You would then invoke the provisioning application to install the DRE in addition to the jclDesktop VM that is already installed:

```
rcplauncher --application com.ibm.rcp.provisioning.application.ProvisioningApplication
 --provisioningOperation provision C:\core.jvm.feature-j2se.xml
```

Alternatively, you can choose to only install the Core JVM Feature – J2SE during the initial installation.

6.1.1 You can update the user or multiuser.xml provisioning manifests to include the feature definition as above (and remove the com.ibm.rcp.jvm.feature definition that refers to the version 1.0.0.x of the jvm feature).

6.1.2 You can update the install.xml provisioning manifest to include the feature definition as above (and remove the com.ibm.rcp.jvm.feature definition that refers to the version 1.0.0.x of the jvm feature).

### Changing to a different runtime and class library

Although you can install both virtual machines (jclDesktop and Java 2 SE 5.0) into the Lotus Expeditor Client, only one VM is active for the platform. To switch the active VM, you must use the Application Management UI to switch the VMs. To do so, select File → Application → Application Management. Expand the /rcp/eclipse site, and locate the Core JVM Feature. Select this feature, then select the task Replace with another version. This will open a dialog prompting you to select another version of the feature. You can switch from jclDesktop to J2SE, or J2SE or jclDesktop using this method.
Understanding the VM packaging

The VM that implements the Java specifications is packaged as a feature.

All VM features are child features of the parent feature com.ibm.rcp.jvm. The major version number represents an entirely different VM provider. For instance, the Lotus Expeditor Client ships with JCL Desktop VM and is major version 1. This scheme allows for seamless upgrade of a VM and also allows the switching of VMs using the update manager.

```plaintext
com.ibm.rcp.jvm.feature_2.0.0.0 includes
   com.ibm.rcp.j2se.win32.x86.feature_1.5.0.SR2
com.ibm.rcp.jvm.feature_1.0.0.0 includes
   com.ibm.rcp.jcl.desktop.win32.x86.feature_6.1.2.<date>
```

The VM properties are located in the VM plug-in file jvm.properties. The launcher dynamically adds these properties to the launch command. This method assures that the VM plug-in defines the properties that it needs.

The –vm property can be used on the command line. However, if you need additional VM properties, you are responsible for adding them.
Lotus Expeditor: Assembling and Deploying Lotus Expeditor Applications
Configuring the platform

This section describes the tasks you can perform to configure the Lotus Expeditor Client on the user’s machine.

**Configuring Java system properties**

Applications might require specific system properties to be set at startup when running the Lotus Expeditor Client 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.

See [“Updating the rcpinstall.properties file” on page 93](#) for detailed information.

You can also specify properties on the command line when you launch the platform. See [“Configuring the platform launcher” on page 44](#) for more information.

**Configuring Virtual Machine arguments**

Applications might require the addition of specific arguments for a Virtual Machine (VM) that implements the Java specifications 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 rcpinstall.properties file, which resides in the `.config` directory of the workspace. (See [“Overriding the workspace directory location” on page 20](#) for more information.).

See [“Updating the rcpinstall.properties file” on page 93](#) for detailed information.

You can also specify VM arguments on the command line when you launch the platform. See [“Configuring the platform launcher” on page 44](#) for more information.

**Configuring native library references**

When using the Lotus Expeditor Client on the desktop, 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.

See [“Updating the rcpinstall.properties file” on page 93](#) for detailed information.

**Configuring Java bootclasspath libraries**

When using the Lotus Expeditor Client on the desktop, 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 93 for detailed information.

### Configuring the platform launcher

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

Options that you might want to configure are as follows:

- **-application**
  Temporarily overrides the application property in the `rcpinstall.properties` file.

- **-config**
  Allows you to specify command options in the `rcplauncher.properties` file to avoid operating system restrictions in the length of commands. See "Updating the rcplauncher.properties file" on page 89 for more information.

  The launcher output is written to a file named `rcplauncher.log` in the logs directory. For a normal launch, it does not contain any output. It contains information if you are using the `-debug` option or if the launcher has a fatal error while launching the platform. In the event of a fatal error prior to the creation of the log, the output is written to `/tmp/rcplauncher.log` on Linux and to `%TEMP%\rcplauncher.log` on Windows.

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

  - If either the `-console` or `-consoleLog` option is specified, then `javaw.exe/expeditorw.exe` specified in the `rcpinstall.properties` file is converted to `java.exe/expeditor.exe` by the `launcher.exe`.
  - `-configuration`: This option is stripped. The configuration location is always calculated as `workspace_location/./config`.

  For a complete list of runtime options defined by Eclipse, see the *Platform Plug-in Developer’s Guide*, which is installed with the Rational® Software Development Platform, and to [http://help.eclipse.org/help32/index.jsp](http://help.eclipse.org/help32/index.jsp)

  To configure the platform launcher to use additional options, you must add the option to the path of the executable. To add options to the executable, modify the shortcut from the desktop icon. Additionally, you can add options as properties in `rcplauncher.properties`. See "Updating the rcplauncher.properties file" on page 89 for more information.

- **-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.

- **-data**
  Temporarily overrides the workspace location (`osgi.instance.area` property). The default calculation of the workspace is specified by the `rcp.data` property in `rcplauncher.properties`. The configuration area is calculated based on this value. A `-configuration` option is ignored.

  **Note:** If the default workspace is not available or inaccessible, specify the workspace during IBM Lotus Expeditor startup using the following command:

  ```
  rcplauncher.exe -data workspace
  ```

  where `workspace` is the path to the desired workspace location.
-debug
Causes additional debugging information to be logged automatically.

-dir
Overrides the bidi direction. Correct values are rtl and ltr.

Note: Overriding the settings to specify rtl is not supported on Linux.

-nl
Overrides the locale being used. The appropriate locale features must have been installed to provide for localized content. For more information, see "Installing additional languages content" on page 170.

Note: Overriding the settings to specify Hebrew or Arabic as a locale is not supported on Linux.

-personality
Customizes the look and feel depending on the desired applications. This can be used to override the value specified in the rcpinstall.properties.

-product
Overrides the product feature that is specified by other means.

-version
Prints the version of the rcplauncher to the temporary log and exits. For example: rcplauncher.exe

-vm
Overrides an external Virtual Machine (VM). You must also pass in any vmargs that are required for this VM. The VM pointed to by this option can cause problems when switched after the fact. IBM only supports the VMs packaged with the product.

-vmargs
Allows additional vmargs to be passed to the VM that implements the Java specifications. No checks are performed on the values. This is not an override, but strictly an addition to the vmargs specified in the rcpinstall.properties file.

Dcommands

See the com.ibm.rcp.core.daemon.package javadoc for information about how to use the daemon.

The com.ibm.rcp.core.daemon class defines an interface that must be implemented by an application developer who uses the com.ibm.rcp.core.daemon.command extension. See the Extension Point Schema documentation in Developing applications for Lotus Expeditor for more information.

You can use this extension point when you need to forward a command to a plug-in from a desktop command. To do so, follow these steps:
1. Add the command as an option to the launch command.
2. Run the launch command. Instead of launching the platform, the daemon forwards the complete launch command to the platform.
3. The plug-in that registered the extension has its RcpDCommand.execute() method called.
4. Call the execute method to decode the command line arguments and act on them in some way.

A plug-in that is to receive commands must:
• Contain the com.ibm.rcp.core.daemon.command extension point in its plugin.xml. A complete example can be found in the Extension Point Schema documentation in Developing applications for Lotus Expeditor.
• Implement the com.ibm.rcp.core.daemon.RcpDCommand interface, which is defined in the Extension Point Schema documentation in Developing applications for Lotus Expeditor.
To activate the command, launch the platform with `rcplauncher.exe -my.plugin#extId.my_command`. If the platform is running, the command is forwarded to the plug-in.

If the platform is not running, the command is ignored and the platform is started. If you need to encrypt the command stream sent to the platform, the following property should be added to the `rcpinstall.properties` file:

```
-Ddaemonconnect.provider=com.ibm.rcp.core.internal.connect.encrypt.EncryptedDaemonConnect
```

The daemon thread listens for commands while the platform is running. It examines each parameter for command ID patterns registered using the extension point. See the `com.ibm.rcp.core.daemon.package` javadoc for more information.

The daemon thread recognizes the following pattern for command IDs:

```
-plugin-id#[extension-id.]command-id
```

where the need for the optional [extension-id] element depends on whether the extension ID was specified in the plug-in manifest when the command was registered.

**Example:** Register the extension point in plugin.xml:

```xml
<extension [id="foo"] point="com.ibm.rcp.core.daemon.command">
  <command id="mycmd" class="org.example.MyCommandClass"/>
</extension>
```

where the extension ID is optional. If the extension ID ("foo") is specified, the RcpD examines the received parameters for matches of the form:

```
-my.plugin#foo.mycmd
```

If the extension ID was not specified in the manifest, the RcpD would be looking for:

```
-my.plugin#mycmd
```

Since the entire parameter list is sent to each command, it is possible for each command to have any number of arguments:

```
-my.plugin#mycmd arg1 arg2 arg3...
```

**Example:** It is also possible to register for all command-line notifications, regardless of content, as follows:

```xml
<extension [id="foo"] point="com.ibm.rcp.core.daemon.command">
  <command id="mycmd" class="org.example.MyCommandClass" matchAll="true"/>
</extension>
```

If you register a handler to receive all commands, you should not register it to receive individual commands also.

**Example:** Application’s plugin.xml:

```xml
<plugin>
  <extension point="com.ibm.rcp.core.daemon.command">
    <command class="com.jdmiles.command.MyCommandClass" id="my_command" matchAll="true"/>
  </extension>
</plugin>
```

**Related tasks**
"Updating the rcplauncher.properties file" on page 89
This section describes how you can update one or more elements of a user’s Lotus Expeditor Client platform by modifying the rcplauncher.properties file.

Related information
"Understanding the client file layout" on page 5
When the Lotus Expeditor Client 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 and theme” on page 115
The Lotus Expeditor Client Workbench leverages Eclipse products and Eclipse themes for branding. You should create your own branding plug-in with your product information instead of changing the one shipped with the Lotus Expeditor Client. Create your own branding feature and deploy it as part of your initial install.

Configuring deployment settings
The provisioning system of the client platform provides the feature install and life cycle management. The provisioning system extends the capabilities provided by the underlying Eclipse Update Manager.

The Eclipse Update Manager and the provisioning system provide a number of preferences to control how these functions operate.

Changing deployment preferences
This section describes the deployment preferences you can change.

The following set of preferences are available and managed using Eclipse Preferences. IBM Lotus Expeditor allows many of them to be set by the user using the Install/Update Preferences page. You can set other preferences through manual file updates of the plugin_customization.ini file or through remote management of the org.eclipse.update.core node, the org.eclipse.update.scheduler node, and the com.ibm.rcp.provisioning node. See "Managing Eclipse preferences” on page 104 for more information on remote management of Eclipse preferences.

Preferences for org.eclipse.update.core:

updatePolicyURL
Location of the update policy file. This may be a local file or one accessible through an HTTP (or similar) URL. This preference is available to the user using the Install/Update Preference page.

Default: Not specified

org.eclipse.update.core.historySize
Number of history update files to maintain for the platform.

Default: 100

org.eclipse.update.core.checkSignature
Whether to check for signed feature objects during installation.

Default: true

org.eclipse.update.core.automaticallyChooseMirror
Automatically check Mirrors during feature installation.

Default: false

org.eclipse.update.core.updateVersions
Select the features that the scan for updates will select prior to display. Available values are compatible, equivalent, or major. This preference is available to the user using the Install/Update Preference page.
Preferences for org.eclipse.update.scheduler:

**download**
Search for updates and notify when they are available (false) or download new updates automatically and notify when ready to install them (true). Ignored unless enabled preference set to true. This preference is available to the user using the Install/Update Preferences page.
Default: false

**enabled**
Whether to automatically search for updates.
Default: false

**schedule**
Determines when to search for new updates. Choices are to search on platform start (on-startup) or on a schedule (on-schedule). Ignored unless enabled preference set to true. This preference is available to the user using the Install/Update Preferences page.
Default: on-startup

**hour**
Time of day to search for new updates (time specified in Hh:00 AM|PM). Note that this value is locale dependent, so a setting for US English is not recognized correctly on another locale. Administrators managing platforms in multiple locales will need to set a locale specific setting. This preference is available to the user using the Install/Update Preferences page.
Default: 1:00 AM

**day**
Day of the week to search for new updates. Values are Every day, or Every <dayname>. Note that this value is locale dependent, so a setting for US English is not recognized correctly on another locale. Administrators managing platforms in multiple locales will need to set a locale specific setting. This preference is available to the user using the Install/Update Preferences page.
Default: Every day

Preferences for com.ibm.rcp.provisioning. (These preferences are not available through the preferences user interface. See "Managing Eclipse preferences" on page 104 for more information on how to change these values.):

**feature.history.size**
Specifies the number of old versions of a feature to retain on the file system. The default values will depend on the platform configuration. If the platform configuration (install.configuration) is set to user, the default is 0. If the platform configuration is set to multiuser, the default value is -1, indicating that all feature versions are retained. The default value used on device configurations is 0.

As an example, if the value is set to 0, then installing an updated version of a feature will cause the previous version of the feature to be removed. If the value is set to 1, then when an updated version of a feature is installed, a single previous version will be maintained. If there was more than one version already installed on the platform, then the oldest versions will be removed. Only older versions of the currently enabled feature are removed. Versions greater than the currently enabled feature will not be removed.

**runDeferredActionsOnStartup**
Specifies when deferred actions should run, such as the uninstall of old versions of features. When set to true, the deferred actions will run immediately when the platform starts, and no progress indicator will be displayed. When set to false, the deferred actions will run after the workbench has completed its startup processing, and a progress indicator will be displayed. The default value is false.
updateSiteList
A list of Strings representing update sites separated by the vertical bar (|) character. For example:

The sites in this list are in priority order. The first site in the list will be searched first, then the second site, and so on. A feature found on the first site will be used, even if the same feature with a higher version may be available on the second or third site. The default value for updateSiteList is an empty list.

whiteList
A Boolean specifying how the updateSiteList will be used.

6.1.1 If set to true, then the updateSiteList will be used as a white list of update sites. A white list restricts updates to be retrieved only from the sites on the list. If this preference is set to false, then the updateSiteList is used as the set of default sites, but updates are not restricted to the sites on the updateSiteList. The default value is false.

6.1.2 If this property is set to true, then the Policy URL searching will be turned off and the useList preference, feature.xml update site, and the bookmarks.xml update site will be ignored.

useFeatureURL
A Boolean specifying whether to include update site URLs provided using feature.xml, provisioning manifest, or provisioning API request in the search list.

6.1.1 If set to true, the available update site URLs are used. If set to false, the update site URLs are ignored. The default value is true.

6.1.2 This preference determines if the feature.xml provided update site should be searched. The manifest and provisioning API requests are always used if specified. If the whiteList preference is set to true, then this value will be ignored.

useList
A Boolean specifying whether the list of update site URLs defined in updateSiteList are used.

6.1.1 If set to true, the updateSiteList is used. If set to false, the list of sites is ignored. The default is false.

6.1.2 If set to true, and if the whiteList preference is set to false, then the updateSiteList is used. If the whiteList preference is set to true, then this value will be ignored.

Configuring the update search settings
This section describes how the Expeditor platform provides for automatic searching for updates.

There are several ways that the Expeditor platform provides for automatic searching for updates:

- From the File → Application → Install dialog
- From the File → Application → Application management dialog (when the Lotus Expeditor product is selected)
- From an individual feature in the Application Management dialog, if an update site has been configured for that particular feature
- By enabling Automatic Updates using the Install/Update Preference Page or by programmatically updating the applicable Eclipse preference

The search hierarchy for Automatic Updates or Scan for Updates is as follows:

- Update Policy URL
- Update sites specified in individual features
To prevent any updates from occurring based on update sites in individual features, the update policy should always contain a pattern specified as “*”. Otherwise, if a feature is installed that does not match any patterns in the update policy, the individual update site specified by that feature will be used.

Automatic updates will search for feature updates of equivalent, compatible, or major, based on the preference setting org.eclipse.update.core/org.eclipse.update.core.updateVersions.

Specifying discovery sites and bookmarks for features

Bookmarks are update sites defined in the bookmarks.xml file residing in the configuration directory. Contributions to the bookmarks file can be provided through discovery sites defined in individual features. Bookmarks are not used during the automatic search.

Enabling and disabling the Update user interface

The Expeditor product provides access to the update manager user interface for deployment of new features and managing of the current configuration.

In addition, preference pages are provided to enable configuration of the behavior of the provisioning system, such as the platform update policy or scheduling of automatic updates. If desired, you can hide the menu actions and preference pages by using the activity support of the workbench.

Define the following extension to the org.eclipse.ui.activities extension point to define an activity that includes the menu actions and preference pages. This extension definition can be included in any plugin.xml (or fragment.xml) of any plug-in (or fragment) deployed to the client platform.

```xml
<plugin>
  <extension
    point="org.eclipse.ui.activities">
    <activity
      id="UIEnabler.installupdatemenu"
      name="Install Update Menu"/>
    <activityPatternBinding
      activityId="UIEnabler.installupdatemenu"
      pattern="com\ibm\eswe\installupdate\launcher/management"/>
    <activityPatternBinding
      activityId="UIEnabler.installupdatemenu"
      pattern="com\ibm\eswe\installupdate\launcher/install"/>
    <activityPatternBinding
      activityId="UIEnabler.installupdatemenu"
      pattern="org\eclipse\update\ui\org\eclipse\update\internal\ui\preferences\MainPreferencePage"/>
    <activityPatternBinding
      activityId="UIEnabler.installupdatemenu"
      pattern="org\eclipse\update\ui\org\eclipse\update\scheduler\AutomaticUpdatesPreferencePage"/>
  </extension>
</plugin>
```

Once the activity is defined, you can enable or disable the activity to display or hide the actions and preference pages. This example includes both menu actions and preference pages in the same activity, but the activity can be defined to include only the capabilities that are desired to be controlled, such as only the menu actions. Once the activity is defined, you can enable or disable the activity as needed.

Configuring feature search order

Learn about an enhanced set of capabilities for managing and controlling the search order when locating new or updated features to install.
Deployers face the following two challenges in managing the set of features installed on any particular system:

- To restrict the places a user can go to obtain new or updated features. If a user is allowed to go to any arbitrary update site to download new or updated features, there is a security risk – the integrity of the client platform could be compromised by malicious features.
- To maintain the set of updates listed in features, or referenced in CA XML, could take considerable time. If the update sites listed in the features or in the CA XML were the only means of locating updates, maintaining or correcting this information in the face of such issues as network topology changes, server renaming, third-party references, inaccessible sites from user’s desktop systems, would be problematic.

The client platform provides an enhanced set of capabilities for managing and controlling the search order when locating new or updated features to install. In addition, the platform now provides the concept of a white list – a restricted set of sites from which to obtain new or updated features. The following preferences are used together to provide flexible control over the update sites from which new features are obtained:

- `com.ibm.rcp.provisioning/updateSiteList`
- `com.ibm.rcp.provisioning/useList`
- `com.ibm.rcp.provisioning/useFeatureURL`
- `com.ibm.rcp.provisioning/whiteList`

**Table 2. White list managed settings and their desired outcomes for 6.1.1**

<table>
<thead>
<tr>
<th>useFeatureURL</th>
<th>useList</th>
<th>UpdateSiteList</th>
<th>whiteList</th>
<th>Desired outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>Can be empty,</td>
<td>True</td>
<td>Use explicit feature update site information, if present. If present, the update site must be in UpdateSiteList. Otherwise, provisioning fails. If not present, search the sites in UpdateSiteList. If UpdateSiteList is empty, provisioning fails.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>which will cause provisioning to fail.</td>
<td></td>
<td>Use explicit feature update site information, if present. If not present, search the sites in UpdateSiteList. If UpdateSiteList is empty, provisioning fails.</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>Can be empty,</td>
<td>False</td>
<td>Use explicit feature update site information, if present. If not present, search the sites in UpdateSiteList. If UpdateSiteList is empty, provisioning fails.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>which will cause provisioning to fail.</td>
<td></td>
<td>Use explicit feature update site information, if present. If present, the update site must be in UpdateSiteList. Otherwise, provisioning fails.</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>Can be empty,</td>
<td>True</td>
<td>Use explicit feature update site information, if present. If present, the update site must be in UpdateSiteList. Otherwise, provisioning fails.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>which will cause provisioning to fail.</td>
<td></td>
<td>Use explicit feature update site information, if present. If no explicit feature update site information is provided (in the provisioning request, feature.xml, or using update GUI, for example), provisioning fails.</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>Can be empty but cannot cause provisioning to fail.</td>
<td>False</td>
<td>Use explicit feature update site information, if present. If no explicit feature update site information is provided (in the provisioning request, feature.xml, or using update GUI, for example), provisioning fails.</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>Must not be empty.</td>
<td>True</td>
<td>Ignore any explicit feature update site URL information. Only search the update sites in UpdateSiteList.</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>Must not be empty.</td>
<td>False</td>
<td>Ignore any explicit feature update site URL information. Only search the update sites in UpdateSiteList.</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>Irrelevant</td>
<td>True</td>
<td>Invalid combination</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>Irrelevant</td>
<td>False</td>
<td>Invalid combination</td>
</tr>
</tbody>
</table>
The default configuration of the platform allows updates to be installed from any update site specified in the provisioning request or using the provisioning API: [useFeatureURL = true, whitelist = false, useList = false, updateSiteList = <empty list>]. While this configuration provides the most flexibility, it also provides the least control over from where new features are obtained. In addition, this configuration relies on the users of the platform and the content of the distributed features to locate new and updated features.

To help control the number of sites from where features can be obtained, the updateSiteList can be defined with a set of update sites. In this configuration [useFeatureURL = true, whitelist = false, useList = true, updateSiteList = <nonemptylist>], in addition to the sites defined by the features, updates and new features can also be found on the set of update sites specified by updateSiteList. The update sites specified by the features will take precedence over the update sites specified by the updateSiteList.

A third setting of the configuration is very similar to using the update policy specified by the org.eclipse.update.core/updatePolicyURL preference. In this configuration [useFeatureURL = false, whitelist = false, useList = true, updateSiteList = <nonemptylist>], features will only be searched for on the update sites specified in the update site list. Whereas the update policy directs specific feature identifiers to specific update sites, this allows the update sites to be specified in a list, without requiring features to be obtained from any one of them. Note that in this configuration, if the updateSiteList is empty, no features will be able to be located, and the feature install requests will fail.

6.1.1 In version 6.1.1, to provide the most control over where new and updated features can be installed from, the configuration [whitelist = true, updateSiteList =<nonemptylist>, useList = true, useFeatureURL = true or false] will restrict updates to come from only the set of update sites specified on the updateSiteList. If useFeatureURL is true, then the feature URLs will take precedence over the list of sites in the updateSiteList, but the update sites specified by the features must be specified in the updateSiteList. Note that in this configuration, if the updateSiteList is empty, no features will be able to be located, and the feature install requests will fail.

Note that since there are three Boolean variables, combinations other than those highlighted above are possible, but will result in a configuration in which provisioning requests can never succeed.

In version 6.1.1, in all of these configurations, the update policy (specified by org.eclipse.update.core/updatePolicyURL) can also be used. If the update policy is specified, then any feature update site remapping will occur prior to applying the preferences as defined above. For example, if a feature com.ibm.sample.feature contains a URL that refers to site http://www.ibm.com, and the update policy contains a mapping of ”com.ibm.sample” to http://w3.mycompany.com, the feature search will be directed to http://w3.mycompany.com. If a restrictive configuration [whitelist = true, useList = true] is used, then the updateSiteList should also contain http://w3.mycompany.com.

When searching the update sites, the searches will occur in sequential order, based on the configuration. If useFeatureUrl = true, then the update site associated with the feature (if any) will be the first search location (subject to the other configuration settings). If the feature is found on this site, and it satisfies the match rules contained in the request, then this feature will be used, regardless of any features that may be contained on any other sites. If the feature is not found on this site, then the search will progress down the list of sites specified in the updateSiteList (assuming that useList = true). Sites in the updateSiteList will be searched in the order specified, until a match is found.

For performance reasons, it is therefore recommended that the updateSiteList, if used, be carefully managed to make sure that the update sites contain the features required, and are organized in the best performing order. Unresponsive sites may significantly increase the time required to complete a provisioning request.
To eliminate confusion as to which feature will be found during searches, it is recommended that the update sites either contain different sets of features or that the same versions of all features are maintained across all sites.

The update manager user interface does not search the updateSiteList before displaying features, so it is possible that features will be listed with no updates available when updated features are available to install in the updateSiteList.

In addition, if whiteList is set to true and the feature URLs for all features are not included in the updateSiteList, then the update manager will allow the selection of features that are not allowed to be installed. IBM Lotus Expeditor will process the feature request from update manager and reject it since the URL for the requested feature is not included in the updateSiteList. This can cause confusion since the feature is shown in update manager as accessible but fails to install.

### Configuring feature search order in 6.1.2

Lotus Expeditor provides an enhanced set of capabilities for managing and controlling the search order when locating new or updated features to install.

Deployers face the following three challenges in managing the set of features installed on any particular system:

- To restrict the places a user can go to obtain new or updated features. If a user is allowed to go to any arbitrary update site to download new or updated features, there is a security risk – the integrity of the client platform could be compromised by malicious features.
- To maintain the currency of the installed set of features. Issues such as network topology changes, renamed servers, third party references, and sites that are inaccessible from user’s desktops make this especially challenging. Maintenance through updating of CA XML files can be difficult and time consuming.
- To ensure that features are updated from the correct location. Users may find they have access to a variety of versions of the same features from different sites and may be confused by which site is the correct one to use. Deployers need the ability to specify the order that Expeditor uses to search for updates for installed features.

The client platform provides an enhanced set of capabilities for managing and controlling the search order when locating new or updated features to install. In addition, the platform now provides the concept of a White List – a restricted set of sites from which to obtain new or updated features. The following preferences are used together to provide flexible control over the update sites from which new features are obtained. See [“Changing deployment preferences” on page 47](#) for descriptions of these preferences.

- `com.ibm.rcp.provisioning/updateSiteList`
- `com.ibm.rcp.provisioning/useList`
- `com.ibm.rcp.provisioning/useFeatureURL`
- `com.ibm.rcp.provisioning/whiteList`
- `org.eclipse.update.core/updatePolicyURL`

The combination of these 5 preferences determines the search order and location for installing feature updates. The following table describes how the settings influence the search order.
<table>
<thead>
<tr>
<th>useFeatureURL</th>
<th>useList</th>
<th>UpdateSiteList</th>
<th>whiteList</th>
<th>Policy URL</th>
<th>Desired outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>True/False</td>
<td>True/False</td>
<td>If empty, this causes no features to be found.</td>
<td>True</td>
<td>Set/Not set</td>
<td>Search the manifest feature URL first if it is in the updateSiteList. Next search the sites in updateSiteList. If search request contains a policy URL, whiteList will erase this value to ensure compliance. Log a message that the policy URL was removed. The sites in feature URL and bookmarks.xml will be ignored. If updateSiteList is empty, no features will be found.</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>Can be empty because feature URL and bookmarks.xml are also used to find features.</td>
<td>False</td>
<td>Not set</td>
<td>Search the manifest feature URL first. Search the sites in updateSiteList next. If updateSiteList is exhausted or empty, then search the feature URL site. If request is from Update UI “Search for new features to install” action, search the sites in bookmarks.xml.</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>Irrelevant</td>
<td>False</td>
<td>Not set</td>
<td>Search the manifest feature URL first. Next use the feature URL to search. If request is from Update UI, search sites in bookmarks.xml.</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>Can be empty because feature URL and bookmarks.xml are also used to find features.</td>
<td>False</td>
<td>Set</td>
<td>Search for features using the policy URL if specified in the search request. Search the manifest feature URL next. If not found, search the updateSiteList. Search the feature URL site next. If request is from Update UI “Search for new features to install” action, search the sites in bookmarks.xml.</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>Irrelevant</td>
<td>False</td>
<td>Set</td>
<td>Search for features using the policy URL if specified in the search request. Search the manifest feature URL next. If not found search the feature URL site. If request is from Update UI “Search for new features to install” action, search the sites in bookmarks.xml.</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>Could be empty because bookmarks.xml is used to find features also.</td>
<td>False</td>
<td>Not set</td>
<td>Search the manifest feature URL first. Use the updateSiteList to search for features. If request is from Update UI, search sites in bookmarks.xml.</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>Irrelevant</td>
<td>False</td>
<td>Not set</td>
<td>Search the manifest feature URL first. If request is from Update UI “Search for new features to install” action, use bookmarks.xml only to search for features.</td>
</tr>
</tbody>
</table>
### Table 3. White list managed settings and their desired outcomes for 6.1.2 (continued)

<table>
<thead>
<tr>
<th>useFeatureURL</th>
<th>useList</th>
<th>UpdateSiteList</th>
<th>whitelist</th>
<th>Policy URL</th>
<th>Desired outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>True</td>
<td>Could be empty because policy URL and bookmarks.xml are used to find features also.</td>
<td>False</td>
<td>Set</td>
<td>Search for features using the policy URL if specified in the search request. Search the manifest feature URL next. Search the updateSiteList next. If request is from Update UI “Search for new features to install” action, search sites in bookmarks.xml.</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>Irrelevant</td>
<td>False</td>
<td>Set</td>
<td>Search for features using the policy URL if specified in the search request. Search the manifest feature URL first. If request is from Update UI “Search for new features to install” action, search sites in bookmarks.xml.</td>
</tr>
</tbody>
</table>

The default configuration of the platform allows updates to be installed from any update site specified in the provisioning request or using the provisioning API: [useFeatureURL = true, whitelist = false, useList = false, updateSiteList = &lt;empty list&gt;]. While this configuration provides the most flexibility, it also provides the least control over from where new features are obtained. In addition, this configuration relies on the users of the platform and the content of the distributed features to locate new and updated features.

To help control the number of sites from where features can be obtained, the updateSiteList can be defined with a set of update sites. In this configuration [useFeatureURL = true, whitelist = false, useList = true, updateSiteList = &lt;nonempty list&gt;], in addition to the sites defined by the features, updates and new features can also be found on the set of update sites specified by updateSiteList. The update sites specified by the features will take precedence over the update sites specified by the updateSiteList.

A third setting of the configuration is very similar to using the update policy specified by the org.eclipse.update.core/updatePolicyURL preference. In this configuration [useFeatureURL = false, whitelist = false, useList = true, updateSiteList = &lt;nonempty list&gt;], features will only be searched for on the update sites specified in the update site list. Whereas the update policy directs specific feature identifiers to specific update sites, this allows the update sites to be specified in a list, without requiring features to be obtained from any one of them. Note that in this configuration, if the updateSiteList is empty, no features will be able to be located, and the feature install requests will fail.

To provide the most control over where new and updated features can be installed from, the configuration [whitelist = true, updateSiteList = &lt;nonempty list&gt;, useList = true or false, useFeatureURL = true or false] will restrict updates to come from only the set of update sites specified on the updateSiteList. Note that in this configuration, if the updateSiteList is empty, no features will be able to be located, and the feature install requests will fail.

Note that since there are three Boolean variables, combinations other than those highlighted above are possible, but will result in a configuration in which provisioning requests can never succeed.

In version 6.1.2, in all of the configurations except whitelist=true, the update policy (specified by org.eclipse.update.core/updatePolicyURL) can also be used. If the update policy is specified, then any feature update site remapping will occur prior to applying the preferences as defined above. For example, if a feature com.ibm.sample.feature contains a URL that refers to site http://www.ibm.com, and the update policy contains a mapping of "com.ibm.sample" to http://w3.mycompany.com, the feature search will be directed to http://w3.mycompany.com.
Update manager behavior when useList or whiteList preference is "true":

When the useList or whiteList preference is "true," the Update Manager will automatically add sites specified in the updateSiteList preference to the list of sites to be searched. These sites can be identified in the Update Manager by the "minus sign" that precedes the update site entry. These sites can not be modified or removed but they can be disabled from search by deselecting the site. When the whiteList preference is "true," the updateSiteList sites are the only sites that are available to the user and no bookmark sites will be displayed.

When searching the update sites, the searches will occur in a specific order. If the feature is found on a site, and it satisfies the match rules contained in the request, then this feature will be used, regardless of any features that may be contained on any other sites.

The order of precedence for searching is:
- Policy URL mapped sites (only if the search request has this set and whiteList preference is "false")
- Feature URL from manifest request (only if manifest has a URL set)
- updateSiteList sites in order (only if useList or whiteList preference is "true")
- Feature URL site from feature.xml (only if useFeatureURL preference is "true" and whiteList preference is "false")
- Bookmark sites in no order (only from Update UI "Search for new features to install" action and if whiteList preference is "false")

For performance reasons, it is therefore recommended that the updateSiteList, if used, be carefully managed to make sure that the update sites contain the features required, and are organized in the best performing order. Unresponsive sites may significantly increase the time required to complete a provisioning request.

To eliminate confusion as to which feature will be found during searches, it is recommended that the update sites either contain different sets of features or that the same versions of all features are maintained across all sites.

Configuring signed plug-in verification

This section describes the signed plug-in policy settings used by the Lotus Expeditor Client provisioning system for controlling access to local or remote Eclipse update sites. Your end users access the update sites to upgrade their base offerings and custom plug-in applications.

Because the provisioning system can access code from a local or remote eclipse update site, signing the jar files posted on the update site and verifying them on the client at download time allows the end user to get reliable information about the code they are about to download. This component allows them to identify who published the code and verify that software has not been tampered with or altered since the time it was uploaded to the update site.

The decisions made by the provisioning subsystem when installing jars from an update site are made by a policy engine which uses policy settings defined as Eclipse preferences to make trust decisions. The policy engine that makes the trust decisions is controlled by a set of Eclipse preferences.

By default, support for verifying signed plug-ins is turned off. To configure signed plug-in verification, modify the following Eclipse preferences. See "Managing Eclipse preferences" on page 104 for setting preference information.

<table>
<thead>
<tr>
<th>Table 4. Signed plug-ins preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Eclipse preference</strong></td>
</tr>
<tr>
<td>com.ibm.rcp.security.update/VERIFICATION_LISTENER</td>
</tr>
</tbody>
</table>
Table 4. Signed plug-ins preferences (continued)

<table>
<thead>
<tr>
<th>Eclipse preference</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.rcp.security.update/EXPIRED_SIGNATURE_POLICY</td>
<td>PROMPT/ALLOW/DENY</td>
</tr>
<tr>
<td>com.ibm.rcp.security.update/UNSIGNED_PLUGIN_POLICY</td>
<td>PROMPT/ALLOW/DENY</td>
</tr>
<tr>
<td>com.ibm.rcp.security.update/UNTRUSTED_SIGNATURE_POLICY</td>
<td>PROMPT/ALLOW/DENY</td>
</tr>
</tbody>
</table>

VERIFICATION_LISTENER

This preference setting indicates which Eclipse IVerificationListener implementation will be used by the provisioning system while verifying jar files being provisioned from an Eclipse update site. This subcomponent provides the below two implementations of IVerificationListener interface:

com.ibm.rcp.security.update.DefaultVerificationListener
Set this value for enabling this subcomponent when provisioning is done by launching the platform in headless mode. This can be enabled at install time by adding this preference to the plugin_customization.ini file in the deploy directory of the media kit.

com.ibm.rcp.security.update.ui.PromptVerificationListener
Set this class to implement the user interface to be shown to the end user. Offerings should set this value to when the platform is launched in non-headless mode and it is expected that the end user will make the trust decisions for untrusted code being downloaded by the provisioning system.

EXPIRED_SIGNATURE_POLICY

This preference setting value defines the default behavior for a given IVerificationListener implementation when it encounters a jar file, which is signed, but the certificate used to sign the jar file has expired.

UNSIGNED_PLUGIN_POLICY

This preference setting value defines the default behavior for a given IVerificationListener implementation when it encounters a jar file that is unsigned.

UNTRUSTED_SIGNATURE_POLICY

This setting value defines the default behavior for a given IVerificationListener implementation when it encounters a jar file that is untrusted.

Setting the above policy values to ALLOW or DENY will be interpreted by the IVerificationListener implementation to allow or deny provisioning of features. However, the policy setting of PROMPT will be interpreted by an IVerificationListener implementation based on whether the platform is running in headless mode. For example, the PromptVerificationListener will prompt the users to make the necessary trust decisions while the DefaultVerificationListener treats PROMPT as DENY so that untrusted code never gets provisioned.

For additional information regarding IVerificationListeners and other public APIs related to signed plug-ins, see the Eclipse javadoc for the package org.eclipse.update.core.

Plug-ins can be signed using either Eclipse tools or the java keytool.

Preloading certificates for initial install and provisioning of signed plugins

When a signed plugin is installed, it is verified using certificates in the media kit keystore and the IBM keystore. The IBM keystore contains public IBM certificates which are used to verify IBM signed plug-ins. The media kit keystore contains public certificates used to verify third party plug-ins.

If a third party wishes to provide public certificates (including self-signed certificates) to be used to verify plug-ins at install time, use the following procedure:
1. Create the following keystore file with no password (using keytool or ikeyman if available). **Note:** Use the keytool (or ikeyman if available) that is included with the target install JVM. This can also be done programatically with the target install VM.

Use the following filename for the given target install VM:

**Desktop EE VM (default):**
- .keystore.jks.J9.install

**J2SE VM:**
- .keystore.JCEKS.IBM_J9_VM.install

**Macintosh JVM:**
- .keystore.JCEKS.Java_HotSpot(TM)_Client_VM.install

2. Use keytool (or ikeyman if available) to add certificates to the newly created keystore.
3. Place the keystore file in the `desktop\install\deploy` directory of the Expeditor install.

**Note:** If supporting more than one target install VM, repeat steps 1-3 using a different VM’s keytool (or ikeyman) and a different keystore file name. The deploy directory can support multiple keystores.

## Configuring the Web Container

The Web Container provides a runtime environment for Web applications in which Web applications can be run both disconnected and connected.

### Configuring the Web Container properties

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 5. Java system properties**

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>Java System Property</th>
<th>ConfigurationAdmin key (this column applies to Expeditor 6.1.1 only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Port</td>
<td>0</td>
<td>Defines the port used by the HTTP Service listener to listen for requests.</td>
<td>com.ibm.pwc.webcontainer.port or</td>
<td>http.port</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A value of 0 indicates that a port will be selected at random when the</td>
<td>com.ibm.pwc.webcontainer.port</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>platform starts. A value of -1 indicates that no listener will be started</td>
<td>(deprecated)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>to listen for HTTP requests. This property now supports a list of values.</td>
<td>com.ibm.pwc.webcontainer.port</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The deprecated property does not support a list of values.</td>
<td>(deprecated)</td>
<td></td>
</tr>
<tr>
<td>HTTPS Port</td>
<td>-1</td>
<td>Defines the port used by the HTTPS Service listener to listen for requests.</td>
<td>com.ibm.pwc.webcontainer.port_secure</td>
<td>https.port</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A value of 0 indicates that a port will be selected at random when the</td>
<td>com.ibm.pwc.webcontainer.port_secure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>platform starts. A value of -1 indicates that no listener will be started</td>
<td>(deprecated)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>to listen for HTTP requests. This property now supports a list of values.</td>
<td>com.ibm.pwc.webcontainer.port_secure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The deprecated property does not support a list of values.</td>
<td>(deprecated)</td>
<td></td>
</tr>
<tr>
<td>HTTP Timeout</td>
<td>60 sec</td>
<td>Defines the value used for socket time-outs</td>
<td>com.ibm.pwc.webcontainer.http.timeout</td>
<td>http.timeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>com.ibm.pwc.webcontainer.http.timeout</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(deprecated)</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Default</td>
<td>Description</td>
<td>Java System Property</td>
<td>Configuration Admin key (this column applies to Expeditor 6.1.1 only)</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>HTTP Address</td>
<td>localhost</td>
<td>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 (for example, <a href="http://www.ibm.com">www.ibm.com</a>, 192.168.0.101, localhost).</td>
<td>com.ibm.pvc.webcontainer.http.address or com.ibm.osg.webcontainer.http.address (deprecated)</td>
<td>http.address</td>
</tr>
<tr>
<td>Min HTTP Threads</td>
<td>4</td>
<td>Defines the minimum number of threads to be started to listen for requests. Valid values are in the range of 0 to 63.</td>
<td>com.ibm.pvc.webcontainer.http.minThreads</td>
<td>http.minThreads</td>
</tr>
<tr>
<td>Max HTTP Threads</td>
<td>20</td>
<td>Defines the maximum number of threads to be started to listen for requests. Valid values are in the range of 2 to 63.</td>
<td>com.ibm.pvc.webcontainer.http.maxThreads</td>
<td>http.maxThreads</td>
</tr>
<tr>
<td>Max Keep Alive Connections</td>
<td>20</td>
<td>Use this property to specify the maximum number of concurrent keep alive (persistent) connections across all HTTP transports.</td>
<td>com.ibm.pvc.webcontainer.http.maxKeepAliveConnections</td>
<td>http.maxKeepAliveConnections</td>
</tr>
<tr>
<td>Max Keep Alive Requests</td>
<td>50</td>
<td>Use this property to specify the maximum number of requests that can be processed on a single keep alive connection.</td>
<td>com.ibm.pvc.webcontainer.http.maxKeepAliveRequests</td>
<td>http.maxKeepAliveRequests</td>
</tr>
<tr>
<td>Keep Alive Timeout</td>
<td>20 sec</td>
<td>Use this property to specify the maximum number of seconds to wait for the next request on a keep alive connection.</td>
<td>com.ibm.pvc.webcontainer.http.keepAliveTimeout</td>
<td>http.keepAliveTimeout</td>
</tr>
<tr>
<td>MIME Mapping configuration file</td>
<td>mimetype.properties</td>
<td>Specifies the MIME mapping file to be used by the Web Container.</td>
<td>com.ibm.pvc.webcontainer.mimemap.configfile</td>
<td>n/a - cannot be configured with Configuration Admin.</td>
</tr>
<tr>
<td>Encoding configuration file</td>
<td>encoding.properties</td>
<td>Specifies the character set encoding file to be used by the Web Container.</td>
<td>com.ibm.pvc.webcontainer.encoding.configfile</td>
<td>n/a - cannot be configured with Configuration Admin.</td>
</tr>
</tbody>
</table>
### Table 5. Java system properties (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>Java System Property</th>
<th>ConfigurationAdmin key (this column applies to Expeditor 6.1.1 only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converter configuration file</td>
<td>converter.properties</td>
<td>Specifies the character converters to be used by the Web Container.</td>
<td>com.ibm.pvc.webcontainer.converters.configfile</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>6.1.1</strong> By default the Web Container will use the converter.properties file located in the Web Container plug-in directory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>6.1.2</strong> By default the Web Container will use the converter.properties file located in the META-INF folder of the Web Container plug-in directory. Users can use this property to supply their own character converter file. Users can specify an absolute path or a path relative to the working directory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtual host configuration file</td>
<td>N/A</td>
<td>Specifies the location of the virtual host configuration file. This configuration file can be used to control access to web applications registered with the Web Container by port or hostname</td>
<td>com.ibm.pvc.webcontainer.vhost.configfile</td>
<td>n/a</td>
</tr>
</tbody>
</table>

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 Virtual Host properties (6.1.2)

The Web Container provides administrators with the capability to restrict access to Web applications to a specific port or list of ports. Refer to this section if you need to restrict access to certain Web applications.

Use the `com.ibm.pvc.webcontainer.vhost.configfile` system property to specify a configuration file that assigns a port(s) to Web applications. For example, if Web application A is mapped to port 80, then it can only be accessed from port 80; accessing Web application A from any other port will result in a 404 error.

The configuration file contains a series of application-port mappings. An application can be mapped to a port or a list of ports. An example configuration file would look like this:

```
/com=localhost:80
/com2=*:80
/com3=:**
/com4=localhost:[80,8777,9999]
/com5=[80,8777,8999]
/com6=80
/com7=xyz.com:*
/com8=xyz.com:8777
/com9=xyz.com:[80,8777]
```

- foo is accessible only from localhost on port 80
- foo2 is accessible from any external machine but only on port 80
- foo3 is accessible from any external machine on any port
- foo4 is accessible only from localhost on ports 80, 8777 and 9999
- foo5 is accessible only from localhost on ports 80,8777 and 8999
- foo6 is accessible only from localhost on port 80
- foo7 is accessible from any external machine on any port IF the hostname of the machine the web application is hosted on is xyz.com
foo8 is accessible from any external machine on port 8777 IF the hostname of the machine the web application is hosted on xyz.com
foo9 is accessible from any external machine on ports 80 and 8777 IF the hostname of the machine the web application is hosted on is xyz.com

The following set of rules apply for the Virtual Host configuration file:
• To protect a servlet, non-J2EE Web application (like WebServices dispatcher servlet), you will need to specify the servlet name similar to what you would do for a J2EE Web application that has a context root (for example, to secure the Web services servlet ws specify /ws=[9000, 9050] if you want to restrict Web services access to ports 9000 and 9050).

Channel Framework Configuration File – This file should be used to configure channels, channel chains and channel factories. Refer to “Using the Channel Framework configuration file” on page 66 for more information on the steps involved in configuring the channel framework using system properties and also an example configuration file.

Configuring the Virtual Host properties (6.1.1)
The Web Container provides administrators with the capability to restrict access to Web applications to a specific port or list of ports. Refer to this section if you need to restrict access to certain Web applications.

Use the com.ibm.pvc.webcontainer.vhost.configfile system property to specify a configuration file that assigns a port(s) to Web applications. For example, if Web application A is mapped to port 80, then it can only be accessed from port 80; accessing Web application A from any other port will result in a 404 error.

The configuration file contains a series of application-port mappings. An application can be mapped to a port or a list of ports. An example configuration file would look like this:
/myApplication=9999
/myOtherApplication=[8777, 9999]

The following set of rules apply for the Virtual Host configuration file:
• If a mapping is specified for a given application, the application can only be accessed on the ports specified (for example, /myContentRoot=9000 means application with context root /myContentRoot can be accessed only on port 9000)
• To protect a servlet, non-J2EE Web application (like WebServices dispatcher servlet), you will need to specify the servlet name similar to what you would do for a J2EE Web application that has a context root (for example, to secure the Web services servlet ws specify /ws=[9000, 9050] if you want to restrict Web services access to ports 9000 and 9050).
• By default, if no mapping is specified for an application then requests for that application will be allowed on ALL ports.
• If the context root is specified but no ports are specified then requests for that application will be allowed on ALL ports.

Configuring the Web Container to use a dynamic port
If the value of the com.ibm.pvc.webcontainer.port Java system property 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.

Lotus Expeditor Client uses a dynamic port by default.

Note: This port will be broadcast to all plug-ins that register a
com.ibm.pvc.webcontainer.listener.HttpSettingListener service. For more information about the
the Web Container ports using the HttpSettingListener Service” in Developing Applications for Lotus Expeditor. When using a dynamic port, the range of the port value is 1-65535. When specifying the value of a Web Container port, administrators must make sure that there is no conflict with any other server on the system. For example, administrators should be careful when setting the Web Container port to 80 since port 80 is frequently used by HTTP servers.

Configuring with Configuration Admin

When using the Lotus Expeditor Client on the desktop, 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 204 for more information on how to configure the Web Container using Configuration Admin and the Enterprise Management Agent.

Configuring the Web Container using the Enterprise Management Agent

You can remotely change the configuration of the Web Container using the Enterprise Management Agent.

Note: The following are prerequisites to the configuration procedure:

- ConfigurationAdmin store has already been pre-populated with Web container configuration data.
- The Enterprise Management Agent is registered with the Device Manager server and is ready to go.

Follow these steps 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 Manager server using the Device Manager console application.

1. Click Devices.
2. Select Use New Query and Return anything as your search criteria, and then click OK. The Device Manager 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. Once the job has completed, select the device, then click View Inventory.
10. Then select Management Tree.
11. Select the configuration entry you want to configure and click Submit Job.
12. Click Next. Leave the defaults as supplied in the panel, then click Next.
13. Enter 1 in 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.

**Configuring HTTPS**

When using the Lotus Expeditor Client on the desktop, the secure hypertext transfer protocol (HTTPS) is a communications protocol used to encrypt data between computers over the Web.

HTTPS uses a Secure Socket Layer (SSL) to transfer encrypted HTTP data. Refer to “Configuring SSL for the Web Container” on page 84 for the required steps.

**Configuring the Web Container using the SocketConfigService**

Third-party bundles can use the com.ibm.pvc.webcontainer.service.SocketConfigService service to dynamically configure the Web Container ports at run time. Review this section if you need to dynamically configure the Web Container ports at run time.

Bundles should use a ServiceTracker to track the com.ibm.pvc.webcontainer.service.SocketConfigService object.

The following example shows how to invoke the com.ibm.pvc.webcontainer.service.SocketConfigService startTransport and stopTransport methods from a bundle’s ServiceTracker.

```java
import com.ibm.pvc.webcontainer.service.SocketConfigService;
import com.ibm.pvc.webcontainer.service.SocketConfigException;
...
private BundleContext context = null;
private ServerSocket socket = null;

public MyServiceTracker(BundleContext bc)
{
    context = bc;
}

public Object addingService(ServiceReference reference)
{
    SocketConfigService service = (SocketConfigService) context.getService(reference);
    if(service != null) {
        try {
            // create a ServerSocket
            socket = new ServerSocket(9999);
            // configure Web Container to listen on port 9999
            service.startTransport(socket);
            ...
        } catch (SocketConfigException e) {
            // handle exception
        }
        ...
    }
}

public void removedService(ServiceReference reference, Object service)
{
    if(service != null) {
        // shutdown transport
        service.stopTransport(socket);
    }
}
```
Enabling and configuring the Channel Framework

The Channel Framework Architecture (CFA) is a highly flexible and scalable solution for client and server transports and provides common networking services, protocols, and I/O operations for the Expeditor Web Container.

Taking a protocol stack abstraction to building a transport, individual channels within this architecture may be thought of as protocol layers in a network stack. This architecture provides extended plugability along the entire chain of events involved in handling communication with the server and processing of the content of the communication at various steps in the server.

The CFA defines a set of interfaces that can be used to implement two main types of objects, channels and channel chains. Channels are used to transport data between the network and the Web Container server component. Channels are an encapsulation of the logic for processing some part of a data stream or for interfacing with a component. The data stream may be part of an inbound request to an application server or an outbound request from an application server. Channel chains consist of a set of channels that are linked together and used to transport data from the network to a component or from a component to the network. By defining a standard mechanism for combining channels, it becomes possible to plug in custom channels that support requirements unique to a particular customer or environment.

In the CFA, a component that is at the beginning or the end of a channel chain is known as the channel framework user (CFU). In an inbound channel chain, the request originates at the network and ends at the CFU. In an outbound channel chain, the request originates at the CFU and ends at the network. The Expeditor Web Container is a CFU. CFUs may have many inbound channel chains leading to them and many outbound channel chains leading from them.

Types of channels

Channels are classified into the following types:

- **Connector channels** are those channels closest to the network interface. A connector channel is the lowest channel in the channel protocol stack. An example of a connector channel is a TCP (Transmission Control Protocol) channel.
- **Protocol channels** are responsible for abstracting information that is transferred through them. When reading information, a protocol channel will generally parse the information into high-level structures that map to constructs which are specific to the protocol. When writing information, a protocol channel will take information provided in protocol-specific structures and map it to the structures needed by the channel below it in the stack. There may be zero or more protocol channels in the link between the connector channel and the application channel. An example of a protocol channel would be an HTTP (HyperText Transfer Protocol) channel.
- **Application / Acceptor channels** are the top-level channels. These channels are generally built on top of specific protocols in order to draw the correct data out of them. An example of an application channels is the one included for the Expeditor Web Container.
- **Filter channels** can only appear in the interior of a chain. Filter channels do not perform protocol conversions, and thus should accept and emit the same data type. Filter channels may be used for such applications as logging, SLA enforcement, etc.

The Channel Framework service is responsible for managing the channel framework lifecycle. It is responsible for the correct behavior of channels and channel chains at server startup and shutdown.

Enabling and configuring the Channel Framework using Eclipse extensions

The following steps describe how to enable the Web Container’s Channel Framework using Eclipse extensions.
1. Create an eclipse plug-in and specify the eclipse extensions in the plug-in's plugin.xml descriptor. Refer to the Extension Point Schema documentation in Developing Applications for Lotus Expeditor for more information on the extension that you can specify or configure. A sample plugin.xml descriptor file looks like this.


```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <!-- Register the TCP connector channels -->
  <extension point="com.ibm.rcp.channelframework.service.channel">
    <channel name="TCP_OUT" factory="com.ibm.ws.tcp.channel.impl.TCPChannelFactory">
      <property name="hostname" value="localhost"/>
      <property name="port" value="11111"/>
    </channel>
  </extension>

  <extension point="com.ibm.rcp.channelframework.service.channel">
    <channel name="TCP_IN" factory="com.ibm.ws.tcp.channel.impl.TCPChannelFactory">
      <property name="hostname" value="localhost"/>
      <property name="port" value="11111"/>
    </channel>
  </extension>

  <!-- Register the HTTP protocol channels -->
  <extension point="com.ibm.rcp.channelframework.service.channel">
    <channel name="HTTP_IN" factory="com.ibm.ws.http.channel.inbound.impl.HttpInboundChannelFactory">
      <property name="persistentTimeout" value="5000"/>
      <property name="readTimeout" value="5000"/>
      <property name="writeTimeout" value="5000"/>
      <property name="maxKeepAliveRequests" value="30"/>
      <property name="keepAliveEnabled" value="true"/>
    </channel>
  </extension>
</plugin>
```
<extension point="com.ibm.rcp.channelframework.service.channel">
  <channel name="HTTP_OUT"
</extension>

<!-- Register the SSL channels -->
<extension point="com.ibm.rcp.channelframework.service.channel">
  <channel name="SSL_2"
    factory="com.ibm.ws.ssl.channel.impl.WSSSLChannelFactory"
  />
</extension>

<!-- Register the UDP connector channels -->
<extension point="com.ibm.rcp.channelframework.service.channel">
  <channel name="UDP"
    factory="com.ibm.ws.udp.channel.impl.UDPChannelFactory"
  />
</extension>

<!-- Register the Web Container application channel -->
<extension point="com.ibm.rcp.channelframework.service.channel">
  <channel name="WC_APP"
    factory="com.ibm.ws.webcontainer.channel.WCChannelFactory"
  />
</extension>

<!-- Register the test channel chains -->
<extension point="com.ibm.rcp.channelframework.service.chain">
  <chain name="TEST INBOUND CHAIN"
    type="INBOUND"
    group="com.ibm.pvc.webcontainer">
    <channel name="TCP_IN" order="0" type="CONNECTOR"/>
    <channel name="HTTP_IN" order="1" type="PROTOCOL"/>
    <channel name="WC_APP" order="2" type="ACCEPTOR"/>
  </chain>
</extension>

<extension point="com.ibm.rcp.channelframework.service.chain">
  <chain name="TEST INBOUND CHAIN 2"
    type="INBOUND"
    group="com.ibm.pvc.webcontainer">
    <channel name="TCP_IN_2" order="0" type="CONNECTOR"/>
    <channel name="HTTP_IN" order="1" type="PROTOCOL"/>
    <channel name="WC_APP" order="2" type="ACCEPTOR"/>
  </chain>
</extension>

<extension point="com.ibm.rcp.channelframework.service.chain">
  <chain name="TEST INBOUND CHAIN 3"
    type="INBOUND"
    group="com.ibm.pvc.webcontainer">
    <channel name="TCP_IN_3" order="0" type="CONNECTOR"/>
    <channel name="SSL_2" order="1" type="PROTOCOL"/>
    <channel name="HTTP_IN" order="2" type="PROTOCOL"/>
    <channel name="WC_APP" order="3" type="ACCEPTOR"/>
  </chain>
</extension>

All the SSL properties should continue to be specified in the SSL configuration file of the Web Container. The location of this file is specified using the com.ibm.pvc.webcontainer.ssl.configfile system property. (refer to “Configuring SSL for the Web Container” on page 84).

Note: The SSL channel is ONLY supported on J2SE. It does not work on DesktopEE.
Enabling and configuring the Channel Framework using system properties

Channels, channel factories and channel chains can be configured using the system properties in this section. All system properties must be specified in the Channel Framework configuration file. Refer to “Using the Channel Framework configuration file” on page 68 – the location of this file is specified using the com.ibm.rcp.channelframework.configfile system property.

Table 6. TCP Channel properties

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>Java System Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory</td>
<td>TCP_FACTORY (both Inbound and Outbound)</td>
<td>Responsible for creating or finding a channel.</td>
<td>com.ibm.rcp.channelframework.channel.factory:&lt;TCP_channel_name&gt;</td>
</tr>
<tr>
<td>Hostname</td>
<td>N/A</td>
<td>Host name on which the inbound channel will listen.</td>
<td>com.ibm.rcp.channelframework.channel.hostname:&lt;TCP_channel_name&gt;</td>
</tr>
<tr>
<td>Port</td>
<td>N/A</td>
<td>Port on which the inbound channel will listen.</td>
<td>com.ibm.rcp.channelframework.channel.port:&lt;TCP_channel_name&gt;</td>
</tr>
<tr>
<td>Max Open Connections</td>
<td>Min - 1</td>
<td>The maximum number of connections allowed for an inbound channel</td>
<td>com.ibm.rcp.channelframework.channel.port:&lt;TCP_channel_name&gt;</td>
</tr>
<tr>
<td></td>
<td>Max - 2000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7. HTTP Channel properties

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>Java System Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory</td>
<td>Inbound: HTTP_IN_FACTORY</td>
<td>Responsible for creating or finding a channel.</td>
<td>com.ibm.rcp.channelframework.channel.factory:&lt;HTTP_channel_name&gt;</td>
</tr>
<tr>
<td></td>
<td>Outbound: HTTP_OUT_FACTORY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Persist timeout</td>
<td>30000</td>
<td>Time to wait for additional requests on a socket (ms)</td>
<td>com.ibm.rcp.channelframework.channel.persistentTimeout:&lt;HTTP_channel_name&gt;</td>
</tr>
<tr>
<td>Read timeout</td>
<td>60000</td>
<td>Time to wait for a read to complete (ms)</td>
<td>com.ibm.rcp.channelframework.channel.readTimeout:&lt;HTTP_channel_name&gt;</td>
</tr>
<tr>
<td>Write timeout</td>
<td>60000</td>
<td>Time to wait for a write to complete (ms)</td>
<td>com.ibm.rcp.channelframework.channel.writeTimeout:&lt;HTTP_channel_name&gt;</td>
</tr>
<tr>
<td>Maximum requests per connection</td>
<td>100</td>
<td>Maximum requests allowed on a single HTTP connection</td>
<td>com.ibm.rcp.channelframework.channel.maxKeepAliveRequests:&lt;HTTP_channel_name&gt;</td>
</tr>
<tr>
<td>Keep-alive connection</td>
<td>True</td>
<td>Controls whether or not to default to a persistent connection as opposed to a connection that will close after one request/response exchange.</td>
<td>com.ibm.rcp.channelframework.channel.keepAliveEnabled:&lt;HTTP_channel_name&gt;</td>
</tr>
</tbody>
</table>

Table 8. SSL Channel properties

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>Java System Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory</td>
<td>SSL_FACTORY (both Inbound and Outbound)</td>
<td>Responsible for creating or finding a channel.</td>
<td>com.ibm.rcp.channelframework.channel.factory:&lt;SSL_channel_name&gt;</td>
</tr>
<tr>
<td>Protocol</td>
<td>SSL</td>
<td>Optional. Specifies the protocol to use.</td>
<td>com.ibm.ssh.protocol.&lt;SSL_channel_name&gt;</td>
</tr>
<tr>
<td>Keystore type</td>
<td>JKS</td>
<td>Optional. Specifies the type of keyStore to use.</td>
<td>com.ibm.ssh.keyStoreType.&lt;SSL_channel_name&gt;</td>
</tr>
<tr>
<td>Keystore location</td>
<td>N/A</td>
<td>Mandatory. Specifies the location of keyStore on the file system.</td>
<td>com.ibm.ssh.keyStore.&lt;SSL_channel_name&gt;</td>
</tr>
<tr>
<td>Keystore password</td>
<td>N/A</td>
<td>Mandatory. Specifies the password to open the keyStore.</td>
<td>com.ibm.ssh.keyStorePassword.&lt;SSL_channel_name&gt;</td>
</tr>
<tr>
<td>Truststore type</td>
<td>Fully-qualified path to file on the fileSystem.</td>
<td>Mandatory. Specifies the type of trustStore.</td>
<td>com.ibm.ssh.trustStoreType.&lt;SSL_channel_name&gt;</td>
</tr>
<tr>
<td>Truststore location</td>
<td>Plain-text password for keyStore.</td>
<td>Mandatory. Specifies the location of trustStore on the file system.</td>
<td>com.ibm.ssh.trustStore.&lt;SSL_channel_name&gt;</td>
</tr>
<tr>
<td>Truststore password</td>
<td>Plain-text password for trustStore</td>
<td>Mandatory. Specifies the password to open the trustStore.</td>
<td>com.ibm.ssh.trustStorePassword.&lt;SSL_channel_name&gt;</td>
</tr>
<tr>
<td>Client authentication required</td>
<td>false</td>
<td>Mandatory. Specifies whether or not the client application requires authentication.</td>
<td>com.ibm.ssh.clientAuthentication.&lt;SSL_channel_name&gt;</td>
</tr>
<tr>
<td>Enabled cipher suite list</td>
<td>N/A</td>
<td>Mandatory. Enabled cipher suites.</td>
<td>com.ibm.ssh.enabledCipherSuites.&lt;SSL_channel_name&gt;</td>
</tr>
</tbody>
</table>

Table 9. Webcontainer Application Channel properties

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>Java System Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory</td>
<td>WC_APP_FACTORY (both Inbound and Outbound)</td>
<td>Responsible for creating or finding a channel.</td>
<td>com.ibm.rcp.channelframework.channel.factory:&lt;WC_APP_channel_name&gt;</td>
</tr>
</tbody>
</table>

Table 10. Channel chain properties

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>Java System Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain name</td>
<td>N/A</td>
<td>The name of the chain.</td>
<td>com.ibm.rcp.channelframework.chain.name:&lt;chain_id&gt;</td>
</tr>
</tbody>
</table>
Table 10. Channel chain properties (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>Java System Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain group</td>
<td>com.ibm.pvc.webcontainer</td>
<td>The name of the group the chain belongs to. Should be set to the CPU identifier (e.g., com.ibm.pvc.webcontainer)</td>
<td>com.ibm.rcp.channelframework.chain.group.&lt;chain_id&gt;</td>
</tr>
<tr>
<td>Channel list</td>
<td>N/A</td>
<td>List of channels in the chain (separated by ',' and specified in order starting with connector channel)</td>
<td>com.ibm.rcp.channelframework.chain.channelList.&lt;chain_id&gt;</td>
</tr>
<tr>
<td>Chain type</td>
<td>N/A</td>
<td>The type of chain (inbound or outbound)</td>
<td>com.ibm.rcp.channelframework.chain.type.&lt;chain_id&gt;</td>
</tr>
</tbody>
</table>

Table 11. Channel Factory properties

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
<th>Java System Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory class</td>
<td>N/A</td>
<td>The factory class. Must be a valid classname, class package must be on the bundle classpath</td>
<td>com.ibm.rcp.channelframework.factory.class.&lt;factory_name&gt;</td>
</tr>
</tbody>
</table>

Using the Channel Framework configuration file

The following steps describe how to configure the Web Container’s Channel Framework support.

Create the Channel Framework configuration file. Refer to “Enabling and configuring the Channel Framework using system properties” on page 67 for more information on properties that can be specified or configured. The location of the Channel Framework configuration file is specified using the com.ibm.rcp.channelframework.configfile property.

The sample Channel Framework configuration file below shows how to configure two separate HTTP / HTTP channel stacks:

```
# factory properties
com.ibm.rcp.channelframework.factory.class.TCP_TEST_FACTORY=com.ibm.ws.tcp.channel.impl.TCPChannelFactory
com.ibm.rcp.channelframework.factory.class.WC_TEST_FACTORY=com.ibm.ws.webcontainer.channel.WCChannelFactory
com.ibm.rcp.channelframework.factory.class.SSL_TEST_FACTORY=com.ibm.ws.ssl.channel.impl.WSSSLChannelFactory

# channel properties for TCP channel(s)
com.ibm.rcp.channelframework.channel.factory.TCP_TEST=TCP_TEST_FACTORY
com.ibm.rcp.channelframework.channel.port.TCP_TEST=9999
com.ibm.rcp.channelframework.channel.hostname.TCP_TEST=localhost
com.ibm.rcp.channelframework.channel.factory.TCP_TEST_2=TCP_TEST_FACTORY
com.ibm.rcp.channelframework.channel.port.TCP_TEST_2=12222
com.ibm.rcp.channelframework.channel.hostname.TCP_TEST_2=localhost

# channel properties for HTTP channel(s)
com.ibm.rcp.channelframework.channel.factory.HTTP_TEST=HTTP_TEST_FACTORY
com.ibm.rcp.channelframework.channel.persistTimeout.HTTP_TEST=7500
com.ibm.rcp.channelframework.channel.readTimeout.HTTP_TEST=7500
com.ibm.rcp.channelframework.channel.writeTimeout.HTTP_TEST=7500
com.ibm.rcp.channelframework.channel.keepAliveEnabled.HTTP_TEST=true
com.ibm.rcp.channelframework.channel.factory.HTTP_TEST_2=HTTP_TEST_FACTORY
com.ibm.rcp.channelframework.channel.persistTimeout.HTTP_TEST_2=7500
com.ibm.rcp.channelframework.channel.readTimeout.HTTP_TEST_2=7500
com.ibm.rcp.channelframework.channel.writeTimeout.HTTP_TEST_2=7500
com.ibm.rcp.channelframework.channel.maxKeepAliveRequests.HTTP_TEST_2=7500
com.ibm.rcp.channelframework.channel.keepAliveEnabled.HTTP_TEST_2=true

# channel properties for SSL channel(s) (remaining SSL properties including keystore, truststore # locations are defined in the SSL config file)
com.ibm.rcp.channelframework.channel.factory.SSL_TEST=SSL_TEST_FACTORY

# channel properties for Webcontainer application channel(s)
com.ibm.rcp.channelframework.channel.factory.WC_TEST=WC_TEST_FACTORY
com.ibm.rcp.channelframework.channel.factory.WC_TEST_2=WC_TEST_FACTORY

# chain properties
com.ibm.rcp.channelframework.chain.name.1=TEST HTTP CHAIN
com.ibm.rcp.channelframework.chain.group.1=com.ibm.pvc.webcontainer
```
All the SSL properties should continue to be specified in the SSL configuration file of the Web Container. The location of this file is specified using the `com.ibm.pvc.webcontainer.ssl.configfile` system property. (refer to “Configuring SSL for the Web Container” on page 84).

- Launch the Lotus Expeditor Runtime. The Web Container will then proceed to initialize the Channel Framework provider and use the provider to configure each channel specified in the configuration file and subsequently enable the channel chains also specified in the configuration file. All SSL properties will also be provided to the Channel Framework provider.

- Access any registered and active web application on the specified ports.

### Configuring the Portlet Container

The Portlet Container provides a runtime environment for JSR 168 portlet applications in which JSR 168 portlets can be instantiated, used and finally destroyed.

The Portlet Container allows users and developers alike to access and run portlet applications either disconnected (offline) or connected (online).

No configuration of the Portlet Container is necessary.

The Portlet Container is an additional extension processor which leverages Web Container functionality and reuses the Web Container settings (server address and server port, for example). See “Configuring the Web Container” on page 58 for more information.

You can change the images and style sheet used by the portlet container when rendering portlets. See “Configuring the Portlet Container branding” on page 122 for more information.

### Configuring portlet preferences

You can remotely manage and update the default set of portlet preferences.

The Portlet Container uses Eclipse Preferences to persistently store modified and updated portlet preferences. See “Managing Eclipse preferences” on page 104 for more information on how to remotely manage Eclipse preferences.

### Configuring mobile 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.
**Installing Web Services Resource Framework components**

Web Services Resource Framework plug-ins are packaged with the Lotus Expeditor Client installation media but are not installed as part of the default installation of the client. You need to manually install the Web Services Resource Framework components from the Lotus Expeditor Client CD.

To manually install the Web Services Resource Framework components, follow these steps:
1. Insert the Lotus Expeditor Client CD.
2. Start the Lotus Expeditor Client.
3. Click File → Application → Install.
4. Select Search for new features to install and click Next.
5. Click the New Local Location button and browse to the /desktop/updates/platform/ directory on the media or the place where you have unzipped the product zip file. Click Open.
6. Click OK on the Edit Local Location dialog box and then click Finish. The Update Manager will search for the updates.
7. A Search Results screen opens. From the features to install list, expand /updates/platform → runtimes and select Web Services Resource Framework 6.1.0.0. Click Next.
8. Accept the license, and Click Next.
9. Follow the rest of the steps to complete the install.

**Configuring Apache Axis Web services clients**

Apache Axis is an open source implementation of the SOAP. It has been enable to provide runtime support for the JAX-RPC based clients in Lotus Expeditor.

**Configuring Apache Axis client proxy settings**

Apache Axis client runtime requires that you set VM properties when connecting through a HTTP proxy.

Apache Axis client runtime requires the following JVM properties to be set when connecting through a HTTP proxy. Apache Axis client does not recognize the proxy settings provided by the Eclipse preferences in the Expeditor environment.

- `-Dhttp.proxyHost` -- Hostname of proxy server
- `-Dhttp.proxyPort` -- Port on server of proxy
- `-Dhttp.proxyUser` -- Optional username for proxy authentication
- `-Dhttp.proxyPassword` -- Optional proxy server password

**Configuring Managed Settings**

The managed settings framework allows you control over the runtime behavior of client applications by letting you set the values of the settings that the applications are reading out of the Eclipse preference store.

If you also designate these settings as read-only, any changes that the client applications make to the settings will either be prevented or be subsequently overwritten with your values, depending on how the client accesses the settings. If the client program knows about managed settings and accesses them through the ManagedSettings Eclipse preference scope, any changes to read-only settings will be prevented. For other plug-ins, which access the settings in the traditional way, changes can occur but will be later overwritten with your values. Updates are run regularly to add any of your new settings or settings changes to the preference store on the client.

The managed settings framework can retrieve policy, preferences, or any other name and value pairs from any back-end system for which there exists a managed settings provider. Providers contain the knowledge of the specific back-end system. Lotus Expeditor Client includes providers to retrieve settings
both from Portal Policy Manager and from an XML file residing on an http server. If there is no provider available for your back-end system, you may write your own. See "Adding a managed settings provider" on page 135 for more information.

You can manage client settings by completing the following procedures:
1. Specifying settings
2. Configuring providers.
3. (Optional) Setting the managed settings update schedule.
4. Ensuring that the managed settings plug-in gets started.

**Specifying settings**

You can specify the desired values of client setting using the back-end administration system of your choice.

If your back-end administration system is Portal Policy Manager, you can use the Policy Type Editor for Eclipse Preferences to set the values. See "Managing Rich Client Preferences using the Policy Type Editor for Eclipse Preferences" on page 234 for more information. All values from the Portal Policy Manager will be treated as if they were specified as read-only. If your back-end administration system is Domino, you will specify your settings in the policy subsystem of the Domino Administration client.

Eclipse preferences are accessed using a qualifier (typically plug-in name), and optionally a scope, which determine the storage location of the preference. To have your settings values read by the client applications, they need to be stored in the same place that the client is looking for them. There are two possible ways to determine this information:

- Look in the application’s documentation to see if a list of settings has been published or request a list from the application developer.
- Figure out the names of the qualifiers and settings yourself by changing values in the preference panels and other user interface controls provided by the application and examining the resulting changes in the Eclipse Preference store.

The managed settings framework enables administrators to control settings in the following scopes:

- Configuration – Preferences that are stored per installation of the platform. They are shared between workspaces.
- Instance – Preferences that are stored per workspace, or per running instance of the platform.
- It may be possible for custom scopes to be managed as well, depending on their implementation.

ManagedSettingsScope is a custom scope, for use by the managed settings framework and managed settings-aware applications. Its data is obfuscated to discourage end-user tampering. Applications which access their settings using this scope instead of using the standard Eclipse scopes, will be prevented from changing read-only settings. Read-only settings are stored in the managed settings scope as well as the associated standard scope.

A Setting Group is a group of related settings. When the managed settings framework retrieves a setting from a back-end system, the setting group it belongs to is the unit that is retrieved, not the individual setting. The structure of a settings group depends on the back-end system that generates the settings contained by it and the implementation of the provider associated with that back-end system. For example, for the Portal provider, a setting group is equivalent to a policy type.

If you are specifying managed settings in the Portal Policy Manager using the Policy Type Editor for Preferences, there are separate attributes for scope and qualifier that you can set explicitly. For instructions, see "Managing Rich Client Preferences using the Policy Type Editor for Eclipse Preferences" on page 234. For this provider, a setting group maps to the PreferenceSet, which maps to the policy
name. When setting up your preference sets, it is important for deletion tracking that all settings with the same qualifier go in the same preferenceSet. All settings that are retrieved from the Portal Policy Manager will be treated as read-only.

If you are specifying managed settings with the XML File provider, you should make the setting group name match the qualifier (usually plug-in ID). If the scope is not instance scope, you should prepend the name of the scope plus a forward slash character (/) to the name of the setting group.

For example:

```xml
<settingGroup name="org.eclipse.ui" lastModDate="20060131">
    <setting name="showIntro" value="false" isLocked="false"/>
</settingGroup>

<settingGroup name="configuration/com.isv.my.plugin" lastModDate="20060131">
    <setting name="use_large_icons" value="true" isLocked="true"/>
    <setting name="sortOrder" value="descending" isLocked="false"/>
</settingGroup>
```

If isLocked is specified as true, the setting will be treated as read-only. For more information about the XML file provider, see ["Configuring the XML file provider"](page 74).

If you are using a different back-end administrative settings system and your environment allows you to control the name of the setting groups (which ever the concept of a setting group happens to be for that system), it is most efficient to name the setting group after the qualifier that will be used to access the settings on the client. If the settings need to be stored in a scope other than instance scope, you should prepend the scope name and a forward slash character (/) to the name.

For example, if you specify a setting group name of org.eclipse.help.ui, the framework stores all the settings for that group in instance scope using a qualifier of org.eclipse.help.ui. If you specify a setting group name of configuration/org.eclipse.core.internal.preferences.osgi, the framework stores all the settings for that group in configuration scope using a qualifier of org.eclipse.core.internal.preferences.osgi.

If it is not feasible to name the setting group to match the qualifier, you can specify the qualifier by prepending it to each setting name, separating the qualifier and setting name with a forward slash character (/). For example, you could name the setting group, All My Eclipse Preferences and name one of its settings, org.eclipse.core.help.base/bookmarks. You could name another setting, org.eclipse.ui/showIntro. As a result, both settings, bookmarks and showIntro are retrieved together as part of the same setting group, but the former is stored with a qualifier of org.eclipse.core.help.base and the latter with a qualifier of org.eclipse.ui.

- Prepending the scope to the setting group name – The scope applies to all the settings in the setting group. For example: configuration/org.eclipse.core.runtime
- Prepending the scope to the setting name – If you prepend the scope to an individual setting, you must also specify the qualifier for the setting, even if it matches the setting group name. For example: configuration/org.eclipse.core.help.base/bookmarks

The following table contains example values of settings specified by an administrator:

<table>
<thead>
<tr>
<th>Setting name</th>
<th>Group/qualifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>use_large_icons</td>
<td>true</td>
</tr>
<tr>
<td>sortOrder</td>
<td>descending</td>
</tr>
<tr>
<td>com.isv.existing.plugin/helper/color_scheme</td>
<td>ocean</td>
</tr>
<tr>
<td>configuration/com.isv.existing.plugin/max_windows</td>
<td>5</td>
</tr>
</tbody>
</table>
The following resulting locations are created in the standard scopes of the Eclipse Preference store based on the administrator’s settings in the table above:

Root
|-- instance
  |-- com.isv.existing.plugin
     |-- use_large_icons = true
     |-- sort_order = descending
  |-- com.isv.existing.plugin.helper
     |-- color_scheme = ocean
  |-- configuration
     |-- com.ibm.existing.plugin
     |-- max_windows = 5

If this naming scheme for specifying scopes and qualifiers is not feasible in your back-end settings system, for instance, if there is a character limit that is too short, you can write your provider so that it translates between the format that you are able to use and the format that is described above.

Note that any forward slash characters (/) in your setting name other than those that delineate the scope and qualifier will be treated as subnodes of the qualifier. If your intent is to have a literal forward slash characters (/) in the setting name, you need to use two forward slash characters (///) instead.

**Limitations on qualifiers**

The following are the limitations on qualifiers:

- It is very important that a qualifier not exist in more than one setting group.
- Qualifier names can not be "instance", "configuration", "default", "ManagedSettings", or any other names that are used as scope names.

**Update behavior**

If you remove settings from the setting group, they will remain on the client but any read-only settings will be changed to readable. If you remove a whole setting group, all the settings in that setting group will remain on the client but any read-only settings will be changed to readable.

If you change the value of a non-read-only setting and then end user has modified the setting, your update will not be applied, unless you also change the setting to read-only.

**Configuring providers**

To plug into the managed settings framework, the back-end management system that populates the Eclipse preference store with the settings for the application must have a corresponding managed settings provider.

A managed settings provider is a plug-in that knows how to pull settings from an associated back-end management system and format them appropriately for the framework to read them. Lotus Expeditor Client provides providers for settings specified by the Portal Policy Manager and for settings that are specified in XML files. If your application gets its preference settings from a back-end system other than WebSphere Portal or an XML file, you can create a custom provider for it. See “Adding a managed settings provider” on page 135

You must configure the file provider associated with settings specified in XML files. For details, see “Configuring the XML file provider” on page 74.

You do not need to configure the provider for the Portal Policy Manager, which is the back-end administrative system used by the WebSphere Portal-administered client. Instead, you must create an account that identifies the WebSphere Portal server and stores authentication information for connecting to it. See “Managing accounts” on page 102 for more information.
Configuring the XML file provider

The XML file provider is responsible for getting the preference settings from an XML file. The file provider needs to have an XML file containing the managed setting data and must know the location of that XML file.

To configure the file provider:
1. Format the XML file as follows:
   - It must contain a <ManagedSettings> element that contains one or more <settingGroup> elements.
   - Each <settingGroup> element must contain one or more <setting> elements.
   - Each <settingGroup> tag must have the following attributes:
     - name – Use the same name as the qualifier (typically plug-in name but can be anything) that its settings are associated with.
     - lastModDate – Specify the date using the java.text.SimpleDateFormat format. The syntax is YYYYMMDDThhmmssZ, where YYYY=year, MM=month, DD=day, hh=hours, mm=minutes, ss=seconds. The values to the right of the T are optional.
   - Important:
     Every change to a setting group must be accompanied by a change to the lastModDate attribute or the new values will not be updated. If no lastModDate is specified, the values are always updated, even if they are not new.
   - Each <setting> tag must have the following attributes:
     - name – Use a name that identifies what the setting does.
     - value – Provide a default value for the setting.
   - Each <setting> tag may also have the following attributes:
     - isLocked – Boolean. The default value is true. If true, the setting is read-only and any changes that a user or application make to the value set by you, the administrator, are prevented or later overwritten. If this attribute is set to false, the administrator’s setting is treated as a default value that can be changed.
     - overwriteUnlocked – Boolean. The default value is false. By default, a setting that is specified as being unlocked will be treated as a default and will not overwrite any existing value on the client. This is to avoid undoing changes that the user might have legitimately made. However, if this setting is set to true, the unlocked value will be overwritten with this new value even if it means clearing the user’s existing value.
   - For example:

     ```xml
     <ManagedSettings>
     <settingGroup name="com.ibm.workplace.mail.policy" lastModDate="20060131">
       <setting name="alwaysEncryptEmail" value="true" isLocked="true"/>
       <setting name="saveCopyinSentFolder" value="false" isLocked="false"/>
     </settingGroup>

     <settingGroup name="desktop_settings" lastModDate="20060131">
       <setting name="logOffAfter" value="30" isLocked="true"/>
       <setting name="autoSaveAfterNMinutes" value="5" isLocked="true"/>
     </settingGroup>
     </ManagedSettings>
     ```

   See the com.ibm.rcp.managedsettings.provider.file plugin to see the schema.
2. Set the initial location of the XML file in the plugin_customization.ini file using a key of com.ibm.rcp.managedsettings.provider.file/URL.
   - For example:

     ```ini
     ```
You can update the plugin_customization.ini file in the install/deploy/ directory on a CD-structure or the one provided with the download applet/portlet using the instructions provided with it.

3. If you would like to subsequently change the URL for the file provider, you may update the existing XML file to add a new setting group of com.ibm.rcp.managedsettings.provider.file and a setting of URL. The next update will run on the existing URL, but the update after that will run with the new URL. If the new URL is unreachable at the time of the update that is setting it, it will not be saved and the original URL will continue to be used. The URL will not be changed until it is updated at a time that the URL is reachable.

**Note:** If the XML file is being hosted by an HTTP server on which authentication has been enabled and there is no associated account set up for the server, the user will be prompted for a name and password the first time the provider runs. Since the provider runs in the background, the user is likely to be confused by this. You can prevent the authentication prompt from displaying by including code in the application that creates an account for this URL during its setup. See “Managing accounts” on page 102 for more information.

**Federation**

Managed settings allows for the simultaneous use of more than one provider.

For instance, you could specify the setting values for one plug-in in the Portal Policy Manager and for another in the XML file provider. However, the potential exists that the same setting group or qualifier could be defined on both the Portal Policy Managed and in the XML File. This is highly discouraged both because of the ambiguity and for efficiency reasons. If you determine that this is happening, you should fix it as soon as possible. To handle this situation, framework forces the results to be deterministic by using the weight of the provider to determine which setting value is applied. The lowest weight "wins". You can determine the default weight of the providers by looking in the ManagedSettingProvider extension point in their plugin.xml files. The default weight of the file provider is set to 300. The default weight of the Portal provider is 200. If necessary, you can change the weight by creating a managed setting for it, using a qualifier of com.ibm.rcp.managedsettings and a setting name of <provider_id (usually plug-in ID)>-weight.

**Configuring the managed settings update schedule**

The framework uses a scheduled background job to call each provider and tell it to retrieve updated settings from its associated back-end systems and then pass them to the framework to update the values stored on the client.

The provider mechanism also includes a mechanism for a provider to inform the framework when it has new updates, instead of waiting for the poll interval to elapse. If the provider you are using has this capability, you do not need to be concerned about this setting. Neither of the providers shipped with the Lotus Expeditor Client have this capability but other providers could.

By default, the scheduled update job runs every 720 minutes (12 hours). To change this value, do one of the following:

- If your settings are being populated by the Portal Policy Manager, use the Administrative Console to edit the value of the com.ibm.rcp.managedsettings/UpdateTimeInMinutes setting.
- If your settings are being populated by the file provider, create a new setting group called com.ibm.rcp.managedsettings and a setting of UpdateTimeInMinutes.
- If your settings are being populated by a different back-end system, create a new name-value pair to set the update interval. Use a qualifier of com.ibm.rcp.managedsettings and a setting of UpdateTimeInMinutes.

Configuring the platform 75
Ensuring that the managed settings plug-in gets started

For your settings to be applied to the client, the plug-in com.ibm.rcp.managedsettings must be started at runtime.

If you are using the Portal-Managed client, this will happen automatically. It will also happen automatically if any of the plug-ins in the application call any of the methods in the managed settings plug-in. However, if you are providing setting values for an application that has no managed settings-aware plug-ins, you will need to ensure that the plug-in gets started. To do so, use one of the life cycle plug-ins. See “Managing life cycle” on page 108 for more information.

Configuring the network layer

Update the polling interval and proxy settings to configure the network layer to your environment.

If the network is down, the network layer will poll the system to determine when the network is back up. The polling interval preference value is com.ibm.rcp.net.http/local_resource_monitor_interval.

See “Managing Eclipse preferences” on page 104 for more information on how to query or modify preference values.

Note: If your system is connected using a device emulator, such as VMWare, or other connections that share your LAN or wireless network connection, the netstatus API would always return true even after you disconnect the LAN cable and wireless. This is because of the existence of the virtual adaptor. If you would like your application to be notified of a true offline state (or plan to disconnect the system), go to Start → My Network Places, right-click on Properties and disable the shared connections. The Netstatus API will return the correct status.

Proxy settings are global, that is, the whole runtime shares the same proxy settings. The HTTP/HTTPS connection is stateless and uses the proxy settings set when the application opens the connection. The proxy settings are defined as eclipse preferences in IBM Lotus Expeditor. For information on managing these eclipse preferences, see “Managing Eclipse preferences” on page 104.

The following Eclipse preferences are defined:

**com.ibm.rcp.net.http.ProxyHost**
Indicates the proxy server host. The String default is null.

**com.ibm.rcp.net.http.ProxyPort**
Indicates the proxy server port. The integer default is 80.

**com.ibm.rcp.net.http.nonProxyHosts**
Indicates the hosts to connect directly to and not through the proxy server. The value can be a list of hosts, each separated by a vertical bar (|). For example, http.nonProxyHosts="www.foo.com|localhost".

**com.ibm.rcp.net.http.IsSecureProxy**
Indicates whether the proxy needs user ID and password. The Boolean default is false.

To configure the secure proxy programmatically, you can use the following example of setting the preference com.ibm.rcp.net.http.IsSecureProxy:

```java
//put proxy information in preference
IScopeContext context = null;
IEclipsePreferences prefNode = null;
try {
    context = new InstanceScope();
    prefNode = context.getNode("com.ibm.rcp.net.http");
} catch (Exception ex) {
    //handle exception
}
```
// put proxy values into preference settings
prefNode.putInt("http.ProxyPort", 80);
prefNode.put("http.ProxyHost", "myProxyHost.ibm.com");
prefNode.putBoolean("http.IsSecureProxy", true);
prefNode.put("http.nonProxyHosts", "http://localhost");

In addition, you must create an account with the username and password for the secure proxy server. It must be configured programatically with the following values:

- Account name = PROXY.ACCOUNT
- Auth type = PROXY.BASIC
- User name = user name for proxy server
- Password = password for above user name

Sample configuration:

```java
AccountsManager accountManager = AccountsManagerFactory.getAccountsManager();
Account account = accountManager.getAccountByName("PROXY.ACCOUNT");
try {
    if (account == null) {
        account = new Account();
        account.setProperty(Account.USER_NAME, user);
        account.getLoginContext().setPassword(passwd); // save the password
        account.setName("PROXY.ACCOUNT");
        account.setType("PROXY.BASIC");
        accountManager.addAccount(account);
    } else {
        account.setProperty(Account.USER_NAME, user);
        account.getLoginContext().setPassword(passwd); // save the password
        account.setName("PROXY.ACCOUNT");
        account.setType("PROXY.BASIC");
        accountManager.updateAccount(account);
    }
} catch (AccountsException e) {
    e.printStackTrace();
}
```

For information about programatically configuring proxy settings, refer to Configuring the proxy settings for Lotus Expeditor in Developing Applications for Lotus Expeditor.

### Configuring the Web Browser Application

You can configure the Integrated Browser Application with a defined Eclipse preference for your environment. For example, you can specify not to display the Web Browser preference page in the IBM Lotus Expeditor Preference dialog.

The following Eclipse preferences are defined:

- **com.ibm.rcp.ui.browser.launcher/showPreferencePage**
  - Specifies whether or not to display the Web Browser preference page in the IBM Lotus Expeditor Preference dialog. The default Boolean value is true.

- **com.ibm.rcp.ui.browser.launcher/allowUserDefinedHomePage**
  - Specifies whether or not to display the "set home page" UI on the Web Browser preference page. The default Boolean value is true.

- **com.ibm.rcp.ui.browser.launcher/admin HomePage**
  - Indicates the home page set by the administrator. The default string value is “about:blank”.

- **com.ibm.rcp.ui.browser.launcher/enableApplet**
  - Specifies whether or not to enable Java2 applet support. The Boolean default is true.

See Managing Eclipse preferences for more information on how to query or modify preference values.
Configuring platform synchronization

Use the Lotus Expeditor Client platform to manage and configure synchronization.

The Synchronization page provides users the ability to quickly view all the synchronizable applications with their synchronization status. Each synchronizable application is displayed in a row with its name, on and off state, priority, last-run time, scope, summary, server, and status. This page also provides users the ability to start and stop synchronization, as well as a quick way to set synchronization schedules and application specific options.

The Synchronization Schedules page is in the platform preferences dialog. It provides users the ability to set two sets of schedules for normal and high-priority applications, respectively. It also provides the synchronization triggers settings applied to all the synchronizable applications.

You must meet these prerequisites to synchronize:
- An available Portal server is configured correctly in platform preferences (see "Managing Rich Clients using the Portal server" on page 205).
- The platform is set to online.

Enabling synchronization and changing priority

If an application that can be synchronized is installed, you can open the Synchronization page from the Launcher to enable or disable synchronization or change priority of an application.

Click on the check box under the Enabled column for an application to enable or disable the synchronization of it.

Click on the check box under the High Priority column for an application to change between High Priority and Normal Priority for an application. You can also do it by right-clicking on the application and selecting (or unselecting) Use the High-Priority Schedule for this Application.

Note: Synchronization will not be presented as an option in the Launcher when there are no items to synchronize. There will be items to synchronize when there is an available Portal server that is configured correctly in platform preferences.

Configuring auto-synchronization schedules and triggers

Open the platform preferences dialog and click on the Synchronization page to configure the auto-synchronization schedules and triggers.

You can check and uncheck the boxes separately to enable and disable the following schedules and triggers:
- Normal-priority synchronization schedule
- High-priority synchronization schedule
- Synchronize when I start the client trigger
- Synchronize when I reconnect after being disconnected trigger
- Synchronize when I go offline trigger
- Synchronize when I shutdown the client trigger

Check Prompt me to have a confirmation dialog prompt to be displayed before the platform performs synchronization. You can also personalize the schedules by changing the times settings, repeat setting, and days of the week settings.
Configuring platform security

The desktop client platform can use either the IBM J9 Virtual Machine (VM) that implements the Java specifications with jclDesktop class libraries or the IBM J9 VM with the Java 2 SE 5.0 class libraries. The security configuration options are different depending upon the VM that is used.

The Lotus Expeditor Client installs the J9 with jclDesktop as the default VM, and Java 2 SE 5.0 can be installed later. Other products that make use of this platform may provision only Java 2 SE 5.0 and not jclDesktop.

When using the Lotus Expeditor Client on devices, DB2e can use SSL while synchronizing with a server by specifying https in the connection URL.

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

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

The following URL is the index page for information on the security aspects of the IBM J2SE 5 VM:

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.

The Lotus Expeditor Client does not support the use of Java 2 Security to grant or prevent code access based upon identity.

Configuring for jclDesktop

The following Ciphers are provided using JCE in jclDesktop:
• AES
• DES
• DESede
• Blowfish
• PBEWithMD5andTripleDES
• PBEWithMD5andDES

The following CipherSuites are supported for SSL Connections:
• SSL_RSA_WITH_3DES_EDE_CBC_SHA
• SSL_RSA_WTH_DES_CBC_SHA
• SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
• SSL_RSA_WITH_NULL_SHA
• SSL_RSA_WITH_NULL_MD5
• TLS_RSA_WITH_AES_128_CBC_SHA

The JAAS component provides a means for principal-based authentication and authorization
The jclDesktop keytool can only be used to create/operate on JKS keystores. You cannot use it to build JCEKS keystores. JCEKS keystores are currently not supported on jclDesktop.

jclDesktop contains some preloaded certificates in the file lib\security\cacerts. This file is preloaded with a number of root CAs and is used for SSL server certificate validation, signed jar verification and midlet verification.

Switching from jclDesktop to J2SE

See “Changing to a different runtime and class library” on page 40 for information about switching from jclDesktop to J2SE.

Keystore considerations when switching VMs

Keystore files that are created by the Java 2 SE 5.0 VM are not compatible with jclDesktop. Likewise, keystore files created with jclDesktop are not compatible with Java 2 SE 5.0. For this reason, IBM Lotus Expeditor will maintain one keystore file per VM. Because of this, when switching VMs, you must reset all the passwords that were stored in the keystore and the keystore password itself. None of the data stored in one VM’s keystore file will be transferred to the other VM’s keystore file. It is important to note that it is possible to have two completely different sets of stored credentials; one accessible only to jclDesktop and one accessible only to Java 2 SE 5.0.

Specifying the default platform login configuration

A login configuration tells the application which LoginModule to use to authenticate users. A LoginModule describes the interface implemented by authentication technology providers.

The Lotus Expeditor Client defines a default configuration that should be used if no alternate configuration is defined by the application.

It also provides the following platform login configurations that you can use:

• SSO-KS – Stacks the SSOLoginModule on top of the KeyStoreLoginModule class to support single sign-on to the keystore.

Follow these steps to specify third-party login configurations:

In the plugin_customization.ini file in the <install dir>/rcp directory, set the value for the following preference:

• com.ibm.rcp.security.auth/loginConfigName – Specifies the name of the Configuration to use by default when performing a platform login using the ILoginContext.login() method. The default value for Lotus Expeditor is SSO-KS.

For example:

com.ibm.rcp.security.auth/loginConfigName=SSO-KS

Configuring SSL for the platform - 6.1.1

If you need to use Secure Sockets Layer (SSL) with a self-signed test certificate or a certificate that does not have a root certificate contained in the default manager keystore, you must import the certificate into the default trust manager keystore.

Expeditor has the default trust manager keystore preconfigured to be the \jre\lib\security\cacerts file, which in preinstalled with the Virtual Machine (VM) that implements the Java specifications.

Export the certificate from the server’s keyStore and import it into the JRE keystore (used by Expeditor’s default trust manager):
1. To export a certificate from another keystore, open a Command Prompt window and change directory to <install_dir>/rcp\eclipse\plugins\<runtime_jre>\jre\lib\security. Use the keytool command-line utility to export the certificate. To do so, type ..\bin\keytool -export -alias <alias> -keystore <keystore file> -file <certificate file>.cer to generate a certificate file. For more information on keystore management, see the VM’s documentation for the java keytool.

2. Import the certificate you generated by typing ..\bin\keytool -import -trustcacerts -alias <alias> -keystore cacerts -file <filename> where <filename> is the absolute path to the certificate previously created.

Expeditor’s VM is configured by default to support SSL using the VM’s cacerts keystore file. To reconfigure the default configuration, set the following system properties. See “Configuring Java system properties” on page 43 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 VM, refer to the following URL:


### Configuring SSL for the platform 6.1.2

If you need to use Secure Sockets Layer (SSL) with a self-signed test certificate or a certificate that does not have a root certificate contained in the default manager keystore, you are either prompted, denied or allowed, depending on how you configure your managed preferences value for com.ibm.rcp.security.jceproxy/ssl.unknowncert.action.

Three possible values are as follows:

**ALLOW**

Pass the SSL connection for sites with untrusted certificates.

**DENY**

fail the SSL connection for sites with untrusted certificates.

**PROMPT**

If Expeditor is running in headless mode, treat PROMPT equal to DENY. If Lotus Expeditor is running with UI enabled, the user is prompted with the following choices:

1. Do not trust this certificate or its certifying authority. Stop the current operation.
2. Trust this certificate for this session; only.
3. Trust this certificate.

The default value is PROMPT. To change this value, you must change the value of the com.ibm.rcp.security.jceproxy/ssl.unknowncert.action Eclipse preference. See “Managing Eclipse preferences” on page 104 for setting preference information.

The Lotus Expeditor trustmanager is configured by default to support SSL using the cacerts keystore file. Additionally, the trustmanager looks into the platform keystore. To reconfigure the default configuration, set the following system properties. See “Configuring Java system properties” on page 43 for information on how to set the properties.

You can specify the following properties for the client:
For more information on setting these properties, and configuring SSL for the VM, refer to the following URL:


### Configuring SSL for the Enterprise Management Agent

Secure Sockets Layer (SSL) connections are based on the existence of digital certificates to promote secure data exchange between server and client. In Lotus Expeditor, the Enterprise Management agent supports both normal and SSL connections between the client and an Expeditor Server. We recommend that you purchase commercial certificates for which public key certificates are already available on the client devices. This greatly simplifies using secure connections since new certificates do not have to be deployed to the clients. However, you may also use self-signed certificates that you create. The procedures for deploying certificates to desktops and devices are different.

#### Configuring for the desktop

The Enterprise Management Agent expects that the server certificate is provided for SSL communication. For instructions on creating the certificate, refer to your Device Manager server documentation.

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

The Enterprise Management Agent checks three locations for the certificate file and accepts the first available location. The Enterprise Management Agent searches the three certificate 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 Device Manager server job.
2. A file named agentServerCert.arm in the path specified by the System Property `user.home`
3. A file named agentServerCert.arm in the path specified by the directory from which you are working, 
   "./".

If the Enterprise Management Agent finds a file in locations #2 or #3, it will set the CM location property to that location.

You can manage the SSL Configuration with the Device Manager server 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.http` - the OSGi SERVICE.PID that identifies the HTTPS Adaptor implementation.
- `SSLEnabler.certloc` - the truststore file path
- `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 Manager server.
Configuring for devices

Although the installation steps are different for various device platforms, you should be able to use the same certificate to support all platforms.

SSL Configuration for Windows Mobile 2003 SE / 5.0, Windows CE 5.0

You must create a certificate and deploy it to both servers and clients.
1. For instructions on creating certificates, refer to Obtaining a certificate in Using Lotus Expeditor Server
   and using the IBM Key Management Utility (ikeyman) tool.

   **Note:**
   - When you create the certificate, the value for the Common Name (cn= value) field must
     match the server address the Enterprise Management Agent uses to connect with the Device
     Manager server.
   - Ensure the dates for which the certificate is valid are correct.
2. To configure the Expeditor Server for SSL communication, refer to Configuring Device Manager for
3. You must distribute the certificate created by ikeyman to the client devices. Use the keytool.exe from
   the Expeditor Client for Desktop to import the certificate into a cacerts file which can then be
   distributed to clients. This file replaces the existing file in the \eclipse\plugins\com.ibm.pvc.wece.device.win32.arm_6.1.2.0-<date>\jre\lib\security folder, so the file should be
   managed to not destroy any certificates previously deployed.
4. Once the certificate has been deployed to the client, the user can open Application
   ManagerPreferences, select the HTTPS option, and fill in the corresponding account information.
5. The user can press the **Test Connection** button to make sure entered information is correct and click
   **Command** → **OK** to connect with Device Manager server.

SSL Configuration for Nokia E90

The Nokia E90 only supports the X.509 version 1 certificate format and PKCS (.p12) extracted format. You
need to take that into account when generating a certificate that needs to work for all supported
platforms.

**Note:** The support for the Nokia E90 is provided as early release code for internal evaluation and testing.
   The support for the Nokia E90 may not be used for production purposes.
1. You must generate a private key pair complying with X.509 version 1 format and extract it as .arm for
   other platforms and .p12 format for E90 only.
2. Refer to Configuring Device Manager for SSL in Using Lotus Expeditor Server to configure the
   Expeditor Server to use SSL communication.
3. The Nokia E90 does not support the PKCS (.p12) format generated by the IBM ikeyman tool directly.
   An alternative is to take the extracted PKCS certificate generated from ikeyman, and then use a
   third-party tool, such as Open SSL or Firefox 2.0 browser, to transform the certificate into the Nokia
   supported format.

   **Note:** Nokia supported PKCS (.p12) is pkcs8-shrounded-key-bag (encrypted keybag) with
   1.2.840.113549.1.12.10.1.2.
4. Put the PKCS certificate on a storage card and open it on the device, or place the certificate browse to
   it using the Nokia Web browser and select it. Enter the certificate password. Make sure there is no
   error while installing it.
5. Once the certificate has been deployed to the client, the user can open eRCP App MgrPreferences, select
   the HTTPS option, and fill in the corresponding account information.
6. The user can press the Test Connection button to make sure entered information is correct and click Command → OK to connect with Device Manager server.

**Configuring client side SSL support**

To access a server that requires client certificate authentication for SSL connections, the Expeditor Client keystore must contain a personal certificate signed by an authority that is trusted by that server.

The client’s self-signed certificate must be in the platform keystore, the JRE keystore, or another keystore that is specified with system properties at runtime:

- `javax.net.ssl.keyStore`
- `javax.net.ssl.keyStorePassword`
- `javax.net.ssl.keyStoreType`

The platform keystore is located in the Expeditor client workspace after the client has been initialized. The password to this keystore is the Base64 encoded version of the password you use to login to your client. The platform keystore created in the jclDesktop environment is not compatible with the platform keystore created in the J2SE environment. It is recommended to access the platform keystore programmatically. More information on how to programmatically access the platform keystore can be found in the Developing Applications for Lotus Expeditor documentation.

The JRE keystore can be found at `<java_home>/lib/security/cacerts`, with a default password of “changeit”.

**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 Lotus Expeditor Client with an appropriate default configuration.

The Web Services runtime does not provide any SSL specific configuration capabilities and relies on the default platform settings.

**6.1.1** Refer to "Configuring SSL for the platform - 6.1.1" on page 80 for more information.

**6.1.2** Refer to "Configuring SSL for the platform 6.1.2" on page 81 for more information.

**Configuring SSL for the Web Container**

This section describes how to configure SSL for the Web Container

When using Lotus Expeditor Client on the desktop, perform the following steps to enable the Web Container to serve requests using a secured socket:

**Note:** The client must be running the J2SE VM (and not jclDesktop) for SSL support. To switch, see "Changing to a different runtime and class library" on page 40.

1. Configure the Web Container.
   a. Create a SSL configuration file and set the following properties

<table>
<thead>
<tr>
<th>Property</th>
<th>value</th>
<th>optional/mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.ssl.protocol</td>
<td>SSL [default], SSLv3, TLS, TLSv1</td>
<td>optional</td>
<td>Specifies the protocol to use</td>
</tr>
</tbody>
</table>
Table 13. SSL configuration properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>value</th>
<th>optional/mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.ssl.keyStoreType</td>
<td>SSO-KS [default], PKCS12</td>
<td>optional</td>
<td>Specifies the type of keyStore to use</td>
</tr>
<tr>
<td>com.ibm.ssl.keyStore</td>
<td>Absolute or relative path to file on the filesystem. If relative path is specified, path must be relative to the current working directory.</td>
<td>mandatory</td>
<td>Specifies the location of keyStore on the file system</td>
</tr>
<tr>
<td>com.ibm.ssl.trustStoreType</td>
<td>SSO-KS [default], PKCS12</td>
<td>optional</td>
<td>Specifies the type of trustStore</td>
</tr>
<tr>
<td>com.ibm.ssl.trustStore</td>
<td>Absolute or relative path to file on the filesystem. If relative path is specified, path must be relative to the current working directory.</td>
<td>mandatory</td>
<td>Specifies the location of trustStore on the file system</td>
</tr>
<tr>
<td>com.ibm.ssl.keyStorePassword</td>
<td>Plain-text password for keyStore</td>
<td>mandatory</td>
<td>Specifies the password to open the keyStore</td>
</tr>
<tr>
<td>com.ibm.ssl.trustStorePassword</td>
<td>Plain-text password for trustStore</td>
<td>mandatory</td>
<td>Specifies the password to open trustStore</td>
</tr>
<tr>
<td>com.ibm.ssl.clientAuthentication</td>
<td>true</td>
<td>false</td>
<td>mandatory</td>
</tr>
</tbody>
</table>

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 the system property `com.ibm.pvc.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.

6.1.1 Example: If the system property is set to 9000, then the SSL configuration file will look as follows:

```properties
com.ibm.ssl.clientAuthentication.9000=true
com.ibm.ssl.trustStorePassword.9000=[xor] /s+47mLdxPk\=
com.ibm.ssl.keyStorePassword.9000=[xor] /s+47mLdxPk\=
com.ibm.ssl.trustStore.9000=../sampleTrust1.jks
```

6.1.2 Example: If the `com.ibm.pvc.webcontainer.port.secure` system property is set to 9000 and you’ve defined a SSL channel SSL_TEST using either the Channel Framework extensions or system properties, then the SSL configuration file will look as follows:

```properties
com.ibm.ssl.clientAuthentication.9000=true
com.ibm.ssl.clientAuthentication.SSL_TEST=true
```

```properties
com.ibm.ssl.trustStorePassword.9000=password
com.ibm.ssl.keyStorePassword.9000=password
com.ibm.ssl.trustStorePassword.SSL_TEST=password
com.ibm.ssl.keyStorePassword.SSL_TEST=password
```
Note: Both the secure transport and channel are configured to use the same keystore and truststore files in this sample. The password for both of these files is 'password'.

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.

3. Export and import the certificate:

   Note: This step should only be performed if Web applications deployed to the Web Container will be using the HTTPSURLConnection class.

   a. Use the keytool command-line utility to create a self-signed certificate for the server (for example, localhost). To do so, open a Command Prompt window and change directory to the directory where you have the key file (for example, c:\temp\ for the file trustkey.jks). Then type keytool -export -alias server -keystore trustkey.jks -file server.cer to generate a server.cer file.

   b. Export the server certificate from the Web Container keyStore and import it into the JRE keystore (used by the Lotus Expeditor Client URL handler) at:

   `$Expeditor_INSTALL_DIR\rcp\eclipse\plugins\$runtime_jre\jre\lib\security\cacerts`

   The $runtime_jre could be one of the following:

   com.ibm.rcp.j2se.win32.x861.5.0.SR2-yyyyymdd
   com.ibm.rcp.j2se.windows.x861.5.0.SR2-yyyyymdd

   where yyyyymmd is the year, month and date.

   To export the server certificate, change directory to `<install_dir>\rcp\eclipse\plugins\<runtime_jre_dir>\jre\lib\security`. Import the certificate you generated by typing keytool -import -trustcacerts -alias server -keystore cacerts -file <filename> where <filename> is the absolute path to the server certificate created in step 3b.

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 webcontainer.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 Apache Derby ISync implementation, there are no ISync-specific SSL configuration steps required. The IBM Apache Derby ISync provider assumes that the client platform has already been configured to support the SSL outbound connections.

6.1.1 Refer to "Configuring SSL for the platform - 6.1.1” on page 80 for information on setting up SSL.

6.1.2 Refer to "Configuring SSL for the platform 6.1.2” on page 81 for more information.

The Apache Derby synchronization codepage configuration isync.encoding values of ASCII, UTF-8, or UTF8 are not supported when using the JCLDesktop Java runtime environment. The default codepage for your client platform will be used if the isync.encoding property is not set and will function as expected. If UTF-8 or ASCII encoding must be used, then the Java 5 runtime environment will be required.
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 Secure Sockets Layer and DB2 Everyplace mobile devices in the DB2 Everyplace Information Center.

**Using SSL from applications**

This section describes using SSL from applications, specifically creating SSL connections to servers and creating SSL sockets from incoming connections.

**Creating SSL connections to servers**

The default platform configuration for SSL for creating connections to servers is handled as described in the following section:

- “Configuring SSL for the platform - 6.1.1” on page 80
- “Configuring SSL for the platform 6.1.2” on page 81

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 platform security” on page 79 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.

You must switch to J2SE to run FIPS. 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 directories:

**For J2SE on Windows:**
```
installdir\rcp\eclipse\plugins\com.ibm.rcp.j2se.win32.x86_1.5.0.version\jre\lib\security
```

**For J2SE on Linux**
```
installdir\rcp\eclipse\plugins\com.ibm.rcp.j2se.linux.x86_1.5.0.version\jre\lib\security
```

Configuring the platform help

Based on your product's requirements, you can either remove the Help menu completely or remove the Help Contents and About menu items under Help using activities.

To remove Help Menu completely, use the following activity:
<extension point="org.eclipse.ui.activities">
  <activity id="activity.eclipse.help.menu" name="hideHelpMenu"/>
  <activityPatternBinding activityId="activity.eclipse.help.menu"
                        pattern="com\ibm\rcp\platform\personality/help"/>
</extension>

To remove the Help Contents menu item under Help:
<extension point="org.eclipse.ui.activities">
  <activity id="activity.eclipse.help.contents" name="hideHelpContents"/>
  <activityPatternBinding activityId="activity.eclipse.help.contents"
                        pattern="com\ibm\rcp\platform\personality/helpContents"/>
</extension>

To remove the About dialog:
<extension point="org.eclipse.ui.activities">
  <activity id="activity.eclipse.help.about" name="hideHelpAbout"/>
  <activityPatternBinding activityId="activity.eclipse.help.about"
                        pattern="com\ibm\rcp\platform\personality/about"/>
</extension>

Configuring the Lotus Expeditor sidebar

Use the sidebar to perform tasks in other applications on the side without distracting from your work in the current application.

The sidebar is located on the left or right side of the client window and contains one or more panels that correspond to installed applications. For more information, refer to Sidebar in Developing Applications for Lotus Expeditor

Disabling menus and menu items

Eclipse Activities (defined through the org.eclipse.ui.activities extension) can be used to hide contributed menus and menu items.

By defining an activity, a menu or menu item is associated with the activity, and unless the activity is enabled, the menu or menu item will not be displayed. To define the activity, you will need to know the pattern that will match the desired menu or menu items. A table listing the Expeditor contributed menus and menu items is provided in Lotus Expeditor top level menus section of Developing Applications for Lotus Expeditor.

For example, to hide the File > Application > Reset menu action, first locate the correct pattern in the table, then define an extension that includes the pattern:
<extension point="org.eclipse.ui.activities">
  <activity id="activity.eclipse.reset.menu" name="hideReset"/>
  <activityPatternBinding activityId="activity.eclipse.reset.menu"
                        pattern="com\ibm\rcp\platform\personality/resetPerspective"/>
</extension>

You can use the same strategy to disable other menus and menu items in Expeditor by specifying the appropriate pattern for that element. For example to remove the File menu, use pattern="com\ibm\rcp\platform\personality/file". To obtain the appropriate IDs to use in the pattern, see IMenuConstants.java in the User Interface and Personality Framework javadoc.
Modifying platform configuration files

The client platform contains several configuration files that may require updates to successfully deploy and run applications.

The following options can be used to modify the properties files:

- Features that are deployed to a client platform can contain code that runs to perform tasks during the life cycle of a feature. This code is known as an install handler. You can use a global installer handler provided by the client platform or you could create your own install handler to modify the properties files. See [Using the global install handler] on page 151.

- You can use tasks provided by the Device Manager server to query and modify specific elements of managed properties files. For the client platform, the rcpinstall.properties, config.ini, plugin_customization.ini, and rcplauncher.properties files are defined by default to be managed by the Device Manager server. You can also enable additional files to be managed by using the com.ibm.rcp.props.dm.managedFile extension point.

If you are using the Portal administered client capabilities to deploy features, the install handler option is the recommended update model. The Portal administered client capabilities do not permit querying of current values of the properties on the client systems.

Updating the rcplauncher.properties file

This section describes how you can update one or more elements of a user’s Lotus Expeditor Client platform by modifying the rcplauncher.properties file.

The rcplauncher.properties file is managed through install handlers. It contains properties that are required by the launcher for global actions.

Without correct values in this file the platform will not start. Any other properties in this file will be ignored. This file should not be modified except through install handlers.

File locations:

- ${rcp.home}/rcp/rcplauncher.exe
- ${rcp.home}/rcp/rcplauncher.properties

A typical rcplauncher.properties file looks like this:

```
rcp.install.id=1156180825640
```

The branding properties (product and osgi.splashPath) are managed from the rcplauncher.properties.

```
osgi.splashPath=platform:/base/../../../rcp/eclipse/plugins/com.ibm.rcp.platform.personality.branding.nl3
branding.n11,platform:/base/../../../rcp/eclipse/plugins/com.ibm.rcp.platform.personality.branding.nl2,
branding.n12,platform:/base/../../../rcp/eclipse/plugins/com.ibm.rcp.platform.personality.branding.nl1
product=com.ibm.rcp.platform.personality.branding.DefaultProduct
```

The calculation used to locate the user’s workspace is externalized in this file.

```
rcp.data=${env.APPDATA}/Lotus/XPD
```

This value can be customized in the installer.

The rcp.install.id property is a unique number generated by the installer. It is used to assure there are no collisions between multiple installs of the platform. This is copied to the user’s new rcpinstall.properties file.

```
provisioning.manifest=file:\/${rcp.home}/rcp/deploy/install.xml
```
The definition of the base install chosen by the administrator is defined in this file. When a new user starts the platform for the first time the configuration is provisioned to this level.

provisioning.manifest.version=1156180825640

This is the version of the provisioning.manifest file being used. The number usually starts with the current rcplauncher.id. When a user starts the platform and the launcher determines that the manifest version is not identical to the level in the user’s rcpinstall.properties a provisioning cycle will be initiated. The level actually provisioned for a user is written to the user’s rcpinstall.properties file by the provisioning component.

install.configuration=user

This is set by the original install and should never be changed. This is copied to the new user’s rcpinstall.properties file.

update.policy.managed=true

This is an internal property and should not be changed. It may be ignored in future versions of rcplauncher.

rcp.base.location=${rcp.home}/rcp/eclipse/plugins/com.ibm.rcp.base_6.1.0.0-20060817

This property points to the base plugin that contains:

- base rcpinstall.properties file – Initial values represent what is in the core set of features. All changes above this level are caused by the running of install handlers.
- startup.jar, eclipse.exe, and launcher.jar located and executed from this location.
- rcplauncher.exe is located here but is copied to <install dir>/rcp/rcplauncher.exe by the plugin install handler.
- Platform.xml and config.ini are here and are used to seed the initial configuration directory.

This property is copied to a new user’s rcpinstall.properties file. From that point forward, it is allowed to get out of sync with the rcplauncher.properties value. That would happen if a user chooses to run with a different version.

provisioning.application=com.ibm.rcp.provisioning.application.ProvisioningApplication

This is an internal property and should not be changed.

jvm.location=${rcp.home}/rcp/eclipse/plugins/com.ibm.rcp.jcl.desktop.win32.x86_6.1.0.0-20060812/jre/bin/j9w.exe

This points to the Virtual Machine (VM) that implements the Java specifications that should be used by new users. When creating a user’s initial configuration, this property becomes the vm property in the user’s rcpinstall.properties. This property and the vm property are managed by feature install handlers. From that point forward, the user is allowed to choose a different VM version or possibly even a different VM. The vm property would be managed by an install handler.

jvm.feature.id=com.ibm.rcp.jcl.desktop.win32.x86.feature
jvm.feature.version=6.1.0.0-20060812
jvm.parent.feature.id=com.ibm.rcp.jvm.feature
jvm.parent.feature.version=1.0.0-20060930

These are internal properties and are managed by install handlers. They are used for correcting the initial platform.xml.

Windows has a 256-byte limit on command length that can be stored in the properties for an icon. If you are customizing launch icons to contain personalities or Dcomammds, for example, this limit can be easily exceeded. As a work-around, you can define the arguments as properties in the rcplauncher.properties file. The launch command will only have to contain the following:
rcplauncher.exe –config launchconfigone
rcplauncher.exe –config launchconfigtwo

When the launcher notices the -config launchconfigone arguments, it substitutes the properties stored in the rcplauncher.properties file as if they had been included on the command line.

For example, if the actual command should be rcplauncher -console -application
com.ibm.MyApplication -nl de -path "path with spaces and {DBCS}" -vmargs -Dmyproperty=test, the following properties would be stored in rcplauncher.properties:

```plaintext
config.launchconfigone.1 = console
config.launchconfigone.2 = application
config.launchconfigone.3 = com.ibm.MyApplication
config.launchconfigone.4 = nl
config.launchconfigone.5 = de
config.launchconfigone.6 = -path
config.launchconfigone.7 = path with spaces and \uxxx\uxxx...
config.launchconfigone.8 = -vmargs
config.launchconfigone.9 = -Dmyproperty=test
```

Where a space would exist between elements on a full command line, the elements would be represented as separate config arguments. In the example above, the elements -nl de are separated by a space on the command line, so would become separate config arguments.

**Notes:**
- The property file does not ensure that these arguments will stay in order. The numbering scheme assures that the arguments can be restored correctly. The number must start with 1 and be numbered consecutively.
- The path must be entered on the command line surrounded with quotation marks. Not only is that not required to be in the file, it is not allowed. Quotation marks can be added if they are part of the arguments but do not add additional quotes.
- The path can contain non-ASCII or non-ISO8859-1 characters. Any non-ASCII characters must be escaped with \uxxx. You can generate these values with a Unicode 16 editor and converting with the Java program native2asci.
- The vmargs must be entered in the rcpinstall.properties as vmarg.xarg=-Xarg. However, for this file, enter the properties only as described above. The value is always exactly as would be contained on the command line – except as noted.
- Using the –config argument does not restrict adding additional arguments on the actual command line. The launcher does not guarantee the order of combining the properties and the command line arguments (except for the overrides noted below). The launcher will maintain the order of the specified properties from the file.

Some of the arguments that can be entered on the command line or in the config.xxx properties have an override priority. The VM, application, product, personality, and plugincustomization properties have the following override priority:

1. Command line
2. config.xxx property
3. Property in rcpinstall.properties

The workspace has the following priority:

1. –data on the command line
2. –data in the config.xxx property
3. rcp.data property in the rcplauncher.properties

The locale has the following priority:
1. `-n1` on the command line
2. `-n1` in the config.xxx property
3. `com.ibm.rcp.core.locale` in rcpinstall.properties. This is normally set from a preference page and will replace the value set by Priority 4
4. `com.ibm.rcp.core.locale` in rcplauncher.properties. This is normally set by the installer and will populate the default language for a user’s rcpinstall.properties file.

5. The locale calculated by the platform

The BIDI value has the following priority:
1. `-dir` on the command line
2. `-dir` in the config.xxx property
3. `com.ibm.rcp.core.bidi` in rcpinstall.properties. This is normally set from a preference page.
4. Directory calculated by the platform when `-n1` is specified

Note that this file may be populated with config properties from an install handler.

The installer creates and populates the rcplauncher.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. Administrators can add config properties.

For information on how to manage this file, see the following sections:

**Managing using the Device Manager server**
[“Managing properties files” on page 203](#)

**Managing Rich Clients using the Portal server**
[“Managing properties” on page 237](#)

**Managing using another management system**
[“Managing properties” on page 238](#)

To enable the flexibility to change the drive that IBM Lotus Expeditor is launched from, the rcpinstall.properties should not contain absolute path references to files or directories in the install directory.

Rather than specifying the absolute location of the install directory, the string `${rcp.home}` is used to reference the install directory, and `${rcp.data}` is used to refer to the workspace directory. Upon launch, the launcher will replace the value of these tokens with the correct location of the IBM Lotus Expeditor install directory and workspace directory prior to passing the arguments to the Java executable for IBM Lotus Expeditor.

When adding new properties to rcpinstall.properties, or making changes to existing properties, it is important that `${rcp.home}` or `${rcp.data}` is used when referring to the install or workspace directories, and to not use any absolute path references.

The rcp.data property is written by the installer at install time and should not be modified. The rcp.data property gives a versatile way to specify where the workspace is to be located. See[“Installing the Lotus Expeditor Client for shared launching from a network drive” on page 19](#) for information on how to specify this value before installing. This value will be copied to each user’s rcpinstall.properties and will override the default algorithm for calculating the workspace location.

The default value for rcp.data for IBM Lotus Expeditor is `rcp.data=${env.APPDATA}/Lotus/XPD`

If you are upgrading from IBM Lotus Expeditor 6.1, the default rcp.data will keep the original algorithm. For Windows, it appears as:

`rcp.data=${env.HOMEDRIVE}${env.HOMEAPPDATA}/IBM/RCP/${prop.rcp.install.id}/${env.USERNAME}/`
The placeholder `${env."system environmental variable"}` is used to specify the use of a system environmental variable.

The placeholder `${prop."rcplauncher_property"}` is used to specify the use of a property from rcplauncher.properties.

The installer can also write the following properties to the rcplauncher.properties:

- com.ibm.rcp.core.locale
- com.ibm.rcp.core.bidi

For more information on these properties, see "Updating the rcpinstall.properties file." These properties allow the installer to specify a default language for each user other than English. These will be copied to each user's rcpinstall.properties when their configuration is created.

## Updating the rcpinstall.properties file

This section describes how you can update the rcpinstall.properties file.

The rcpinstall.properties file resides in the location `${rcp.data}/.config`. See [Overriding the workspace directory location](#) on page 20 for more information.

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. See the javadoc for java.util.Properties.load() for more information.

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, there is no guarantee which property will be used.

### Table 14. rcpinstall.properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vm=&lt;VM executable file&gt;</td>
<td>This property is required. It must point to an executable file of the Virtual Machine (VM) that implements the Java specifications. The preference is expeditorw, javaw, or ibmjav. You can override this for the user and multi-user configurations by using the -vm argument on the command line. See <a href="#">Changing the virtual machine</a> on page 39 for more information.</td>
</tr>
<tr>
<td>rcp.install.id=&lt;id&gt;</td>
<td>This property is required. It is a unique ID that the installer creates for each installation.</td>
</tr>
<tr>
<td>Xbootclasspath.prepend=&lt;path&gt;</td>
<td>This property specifies the <code>-Xbootclasspath/p:path</code>. See the documentation for the Java application launcher.</td>
</tr>
<tr>
<td>Xbootclasspath.append=&lt;path&gt;</td>
<td>This property specifies the <code>-Xbootclasspath/a:path</code>. See the documentation for the Java application launcher.</td>
</tr>
<tr>
<td>-D&lt;prop&gt;=&lt;value&gt;</td>
<td>See the documentation for the Java application launcher. Additional System properties may be added at the bottom of the file.</td>
</tr>
<tr>
<td>library.path.append=&lt;path&gt;</td>
<td>Modifies PATH (on Windows) or LD_LIBRARY_PATH (on Linux)</td>
</tr>
<tr>
<td>library.path.prepend=&lt;path&gt;</td>
<td>Modifies PATH (on Windows) or LD_LIBRARY_PATH (on Linux)</td>
</tr>
</tbody>
</table>
Table 14. rcpinstall.properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>application=&lt;application plugin id&gt;</td>
<td>This is equivalent to the eclipse runtime property eclipse.application. You can override this for the user and multi-user configurations by using the -application argument on the command line.</td>
</tr>
<tr>
<td>product=&lt;product id&gt;</td>
<td>This is equivalent to the eclipse runtime property eclipse.product. You can override this for the user and multi-user configurations by using the -product argument on the command line.</td>
</tr>
<tr>
<td>personality=&lt;personality id&gt;</td>
<td>Allows you to customize the look and feel depending on the desired applications. You can override this for the user and multi-user configurations by using the -personality argument on the command line.</td>
</tr>
<tr>
<td>plugincustomization=&lt;path to customization file&gt;</td>
<td>This file is provided by the initial install. New values may be appended to this file. You can override this for the user and multi-user configurations by using the -plugincustomization argument on the command line.</td>
</tr>
<tr>
<td>install.configuration={multiuser, service, user, userpb, lockdown}</td>
<td>This property is required. It specifies the configuration installed by the installer. After a platform has been installed, this value should not be changed.</td>
</tr>
<tr>
<td>vmarg.&lt;name&gt;=&lt;value&gt;</td>
<td>You can provide VM specific arguments here. The &lt;value&gt; is passed unaltered as a vmarg to the VM. The &lt;name&gt; is only used as a unique identifier of the vmarg within this file. Syntax of the argument is not checked.</td>
</tr>
<tr>
<td>library.preload=&lt;value&gt;</td>
<td>This property is required for Linux. This will have been populated by the installer if needed.</td>
</tr>
<tr>
<td>com.ibm.rcp.core.locale</td>
<td>Optional: This is added by the launcher through public methods. It can be overridden on the command line with -nl localearg.</td>
</tr>
<tr>
<td>com.ibm.rcp.core.bidi</td>
<td>Optional: This is added by the launcher through public methods. It can be overridden on the command line with -dir dir.</td>
</tr>
<tr>
<td>env.set.system env variable</td>
<td>This may be used to set a system environmental variable. If the variable exists, it will be replaced. For instance env.set.XXX=a new variable would set XXX to “a new variable”.</td>
</tr>
<tr>
<td>&lt;any java.util.logging configuration settings&gt;</td>
<td>Optional: Any of the standard java.util.logging configuration settings can be made in this file such as logger level configuration entries and handler listings. For a complete list of java.util.logging configuration values, see “Configuring platform logging and tracing” on page 96.</td>
</tr>
</tbody>
</table>

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.
# Wed Feb 01 08:42:42 CST 2006
# Note: this file has been formatted to aid readability. In the real file order is
# not guaranteed and comments will be stripped.

Xbootclasspath.append=${rcp.home}/rcp/eclipse/plugins/com.ibm.rcp.base_6.1.1.20070222
0600/rcpbootcp.jar
-Dcom.ibm.pwc.webcontainer.port=0
-Declipse.registry.nulltoken=true
-Djava.net.proxyhost=www.protocol
-Djava.net.proxyport=8080
-Dosgi.framework.extensions=com.ibm.rcp.core.logger.frameworkhook,com.ibm.jxesupport
# Eclipse plugin_customization.ini override
plugincustomization=${rcp.home}/rcp/plugin_customization.ini

# Exclude Eclipse logging hook and JXE support hook
-Dosgi.framework.extensions.exclude=org.eclipse.core.runtime.adapters.EclipseLog
Hook,com.ibm.jxesupport.CDSHookConfigurator

# Java security file location
-Djava.security.properties=file:${rcp.home}/rcp/eclipse/plugins/com.ibm.rcp.base_6.1.1
200702220600/rcp.security.properties

# Java Core dump location. Note this is temp fix for only VM environmental variables
# This ability is not supported in the general case in 6.1
env.set.IBM_JAVACOREDIR=${rcp.data}/logs
env.set.IBM_COREDIR=${rcp.data}/logs
env.set.IBM_HEAPDUMPDIR=${rcp.data}/logs

# JSR47 Logging Configuration
handlers=java.util.logging.ConsoleHandler
com.ibm.rcp.core.internal.logger.boot.RCPLogHandler
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler
com.ibm.rcp.core.internal.logger.boot.RCPLogHandler.encoding=UTF-8
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler.encoding=UTF-8
java.util.logging.ConsoleHandler.encoding=UTF-8
java.util.logging.ConsoleHandler.level=INFO
java.util.logging.ConsoleHandler.level=INFO
com.ibm.rcp.core.internal.logger.frameworkhook.level=INFO
com.ibm.rcp.core.internal.logger.boot.RCPLogHandler.level=INFO
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler.level=WARNING
com.ibm.rcp.core.internal.logger.boot.RCPLogHandler.append=false
com.ibm.rcp.core.internal.logger.boot.RCPLogHandler.pattern=error-log-%g
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler.append=false
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler.append=false
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler.level=FINEST
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler.pattern=trace-log-%g

# RCP Logging Configuration

# RCP Trace Configuration

For information on how to manage the rcpinstall.properties file, see the following sections:

**Managing using the Device Manager server**

"Managing properties files" on page 203

Configuring the platform  95
Managing Rich Clients using the Portal server

“Managing properties” on page 237

Managing using another management system

“Managing properties” on page 238

To enable the flexibility to change the drive that IBM Lotus Expeditor is launched from, the rcpinstall.properties should not contain absolute path references to files or directories in the install directory.

Rather than specifying the absolute location of the install directory, the string ${rcp.home} is used to reference the install directory, and ${rcp.data} is used to refer to the workspace directory. Upon launch, the launcher will replace the value of these tokens with the correct location of the IBM Lotus Expeditor install directory and workspace directory prior to passing the arguments to the Java executable for IBM Lotus Expeditor.

When adding new properties to rcpinstall.properties, or making changes to existing properties, it is important that ${rcp.home} or ${rcp.data} is used when referring to the install or workspace directories, and to not use any absolute path references.

Configuring platform logging and tracing

The Lotus Expeditor Client platform leverages the Java Runtime Environment java.util.logging logging framework for persistence of all of the platform and application log and trace messages.

The default configuration of the platform logging framework is stored in the user’s workspace/.config/ rcpinstall.properties file. For information on updating this file, see “Updating the rcpinstall.properties file” on page 93.

System Default Configuration

Handlers

The default configuration for the client platform registers three handlers: a console handler, a log handler and a trace handler. The encoding for the file-based handlers (log and trace) is set to UTF-8 while the console encoding defaults to the platform encoding method for all log messages. The number of files (count) to be persisted for each of the handlers defaults to 6, and the maximum size (size) for each of the files before they will be rotated defaults to 4000000 bytes. The size can be set individually for both the log and trace handlers and has a minimum size of 100K bytes. The default message level filter for the console handler and the trace handler is FINEST allowing all messages that are logged to be captured by these handlers and processed appropriately, while the log handler defaults to WARNING to ensure that the system log, which is intended for the end user is not overrun with informational messages and trace messages during normal execution. This setting allows the end user to more easily identify possible problems in the log file in the case where the application/platform is not functioning as expected.

One important note is that while the standard java.util.logging settings are used for most of the configuration, the one variation from this is the pattern setting. The pattern for the Lotus Expeditor Client is expected to just be a relative name from the users workspace/logs directory and does not support the %h replacement defined by java.util.logging to relocate these log files.

Formatters

If the platform is started with a console, all log and trace messages are routed to the console handler and displayed using the standard text format using the RCPFormatter. The standard log and trace files are written by default in CommonBaseEvent XML format using the CBE101Formatter.

Logger level configuration
The default rcpinstall.properties file also includes a small collection of default java.util.logging logger level configuration as well. There are 2 special named loggers defined to handle system.err and system.out messages they are called SystemOut and System.err. They default to the INFO level which means that system.err and system.out messages will not appear by default in the system.log file but they will appear in the system trace file.

Default Configuration

The default settings for the java.util.logging framework are shown here:

```java
# JSR47 Logging Configuration
handlers=java.util.logging.ConsoleHandler
com.ibm.rcp.core.internal.logger.boot.RCPLogHandler
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler
com.ibm.rcp.core.internal.logger.boot.RCPLogHandler.encoding=UTF-8
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler.encoding=UTF-8
.level=WARNING
SystemOut.level=INFO
SystemErr.level=INFO
com.ibm.rcp.core.internal.logger.frameworkhook.level=CONFIG
com.ibm.rcp.core.internal.logger.boot.RCPLogHandler.level=WARNING
com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler.level=FINEST
java.util.logging.ConsoleHandler.level=FINEST
java.util.logging.ConsoleHandler.formatter=com.ibm.rcp.core.internal
.logger.boot.RCPFormatter

# RCP Logging Configuration
#com.ibm.rcp.core.internal.logger.boot.RCPLogHandler.formatter=
com.ibm.rcp.core.internal.logger.boot.RCPLogFormatter
com.ibm.rcp.core.internal.logger.boot.RCPLogFormatter
com.ibm.rcp.core.internal.logger.boot.RCPLogFormatter
com.ibm.rcp.core.internal.logger.boot.RCPLogFormatter
com.ibm.rcp.core.internal.logger.boot.RCPLogFormatter
com.ibm.rcp.core.internal.logger.boot.RCPLogFormatter
size=4000000
count=6
append=false
pattern=error-log-9g

# RCP Trace Configuration
#com.ibm.rcp.core.internal.logger.boot.RCPTraceHandler.formatter=
com.ibm.rcp.core.internal.logger.boot.RCPFormatter
com.ibm.rcp.core.internal.logger.boot.RCPFormatter
com.ibm.rcp.core.internal.logger.boot.RCPFormatter
com.ibm.rcp.core.internal.logger.boot.RCPFormatter
com.ibm.rcp.core.internal.logger.boot.RCPFormatter
com.ibm.rcp.core.internal.logger.boot.RCPFormatter
size=4000000
count=6
append=false
pattern=trace-log-9g
```

Log message federation

While the JRE java.util.logging APIs are used for the actual persistence of messages to disk, it is important to note that there are several logging APIs available in the Lotus Expeditor Client platform. The client platform provides the following logging APIs: java.util.logging APIs, eclipse logging APIs, the OSGi LogService APIs, and Apache Commons logging APIs. The core logging framework of the Lotus Expeditor Client platform captures the messages from all of these APIs and federates them into one single log using the java.util.logging persistence and formatting framework. To simplify configuration of these federated messages, dynamic named loggers are created for the non-java.util.logging components when messages of the appropriate level are generated and logged. The named loggers are created with the bundle symbolic name of the OSGi bundle that is logging the message as the name of the java.util.logging logger. This logger can be configured using the standard java.util.logging logger level configuration using the rcpinstall.properties file or dynamically using the OSGi console command setlogrlev. See [Dynamically adjusting the log level](#) in *Lotus Expeditor Troubleshooting and Support* for more information on this capability.
Updating the config.ini file on desktops

This section describes the properties provided in the default config.ini file.

The config.ini file is located in the `<configurationdir>/ .config` directory. The installer provides information and values for this file that the platform needs.

The .config directory resides in the workspace directory defined by the pattern in the rcp.launcher.properties file.

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

**osgi.parentClassloader**

The definition of the parent class loader for OSGi bundles.

- **Current Setting:** ext
- **Default setting:** boot

**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 and to Eclipse [http://help.eclipse.org/help32/index.jsp](http://help.eclipse.org/help32/index.jsp).

- **Current Setting:**
  
  
  ```
  osgi.bundles=org.eclipse.equinox.common@2:start, \ 
  org.eclipse.core.jobs@4:start,\ 
  org.eclipse.equinox.registry@4:start,\ 
  org.eclipse.core.runtime.compatibility.registry,\ 
  org.eclipse.equinox.preferences@4,\ 
  org.eclipse.core.runtime.compatibility.registry,\ 
  org.eclipse.equinox.core@4,\ 
  org.eclipse.equinox.core.contenttype@4,\ 
  org.eclipse.equinox.core.runtime@4:start, \ 
  org.eclipse.equinox.core.runtime@4:start,\ 
  /../rcp/eclipse/plugins/com.ibm.rcp.lifecycle.platform@5: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:** 10
- **Default Setting:** 4

**osgi.startLevel**

Sets the top start level for the framework to attempt to start.

- **Current Setting:** 15
- **Default Setting:** 6

**eclipse.exitOnError**

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

- **Current Setting:** false
- **Default Setting:** true

For information on how to manage the config.ini file, see the following sections:

**Managing using the Device Manager server**

[“Managing properties files” on page 203](http://help.eclipse.org/help32/index.jsp)
Managing Rich Clients using the Portal server

```
Managing properties" on page 237
```

Managing using another management system

```
Managing properties” on page 238
```

**Updating the config.ini file on devices**

On devices, the config.ini file is located in the `\eclipse\configuration` directory. This file can be modified to set Java system properties which are usually not specific to single applications.

See ["Changing a property file value” on page 203](#) for more information on setting properties on users’ devices.

**Updating the plugin_customization.ini file**

This section describes the properties provided in the default plugin_customization.ini file.

This file sets the default value of the preferences used on the system. The plugin_customization.ini is a Java properties file that contains preference settings. Each preference is generally identified by a node name and a key. Some preferences can define nodes between the root node and the key.

The format of each line is the following:

```
root nodename/[optional intermediate node/] preference name=preference value
```

To add a new default preference, add a new line to the file specifying the nodenames, preference name, and value.

For example, to set an update policy file, first identify the node associated with the preference (org.eclipse.update.core), the preference name (updatePolicyURL), and the desired value (http://w3.myserver.com/policy.xml)

```
Then add the line org.eclipse.update.core/updatePolicyURL=http://w3.myserver.com/policy.xml
```

As a Java properties file, content is expected in ISO-8859-1 encoding. Any characters not in the ISO-8859-1 encoding should be specified using Unicode escape syntax. Refer to the javadoc for the java.util.Properties class for more information.

The plugin_customization.ini file resides in the `<install dir>/rcp` directory.

The current settings in the file are the following:

```
com.ibm.rcp.security.auth.ui/ssoEnabled=false
com.ibm.rcp.platform/portalServerAddress=
com.ibm.rcp.platform/portalEnabled=false
com.ibm.rcp.security.auth.ui/ssoAllowed=true
```

In the plugin_customization.ini file in the `<install dir>/rcp` directory, set the values for the following preferences to true:

```
com.ibm.rcp.security.auth.ui/ssoAllowed
```

Determines whether or not users have the option of using single sign-on. You can set this preference value during the client installation or later using a managed setting. This preference should not be surfaced to users.

```
com.ibm.rcp.security.auth.ui/ssoEnable
```

Determines whether or not users have the option of turning single sign-on on or off. If the option is set to true, single sign-on is used. If the option is set to false, single sign-on is disabled. The value of this preference is relevant only if ssoAllowed is true.
For example:
com.ibm.rcp.security.auth.ui/ssoEnabled=true
com.ibm.rcp.security.auth.ui/ssoAllowed=true

Note: The ssoEnabled and ssoAllowed values can only be set before the platform is run. Once the platform has been run, these options can no longer be changed.

For information on how to manage the plugin_customization.ini file, see the following sections:

Managing using the Device Manager server
“Managing properties files” on page 203

Managing Rich Clients using the Portal server
“Managing properties” on page 237

Managing using another management system
“Managing properties” on page 238

Enabling platform single sign-on

Single sign-on (SSO) authenticates users by prompting them for a user name and password a single time. Enabling platform single sign-on gives users secure access to the platform keystore without displaying additional authentication prompts.

To enable platform single sign-on, perform the following step:

In the plugin_customization.ini file in the install_dir/rcp directory, set the values for the following preferences to true:
• com.ibm.rcp.security.auth.ui/ssoAllowed – Boolean value. Determines whether or not users have the option of using single sign-on. You can set this preference value during the client installation or later using a managed setting.
• com.ibm.rcp.security.auth.ui/ssoEnable – Boolean value. Determines whether or not users have the option of turning single sign-on on or off. If set to true, single sign-on is used. If set to false, single sign-on is disabled. The value of this preference is relevant only if ssoAllowed is true.

For example:
com.ibm.rcp.security.auth.ui/ssoEnabled=true
com.ibm.rcp.security.auth.ui/ssoAllowed=true

Note: The ssoEnabled and ssoAllowed values can only be set before the platform is run. Once the platform has been run, these options can no longer be changed.

Managing Secure KeyStore

The Java KeyStore is the standard storage abstraction for security-sensitive information like keys, certificates and passwords.

The Java keystore is accessed by the Login Modules to retrieve and store passwords as part of the login process. The Java KeyStore is also accessed by the Accounts API to change passwords outside of the login process.

The Java KeyStore must be unlocked to access any data stored inside of it. The KeyStore can be unlocked using Platform Login. By default, Lotus Expeditor calls the Platform Login to unlock the keystore at startup time. This can be changed programmatically to unlock the KeyStore when needed.

Here is an example of how to programmatically Call the SecurePlatform login to unlock the Java Keystore:
import com.ibm.rcp.security.auth.SecurePlatform;

public void doLogin() {
    try {
        SecurePlatform.getLoginContext().login();
    } catch (LoginException e) {
        // Login failed... exit platform or take corrective action
    }
}

Depending on how Lotus Expeditor is configured, calling SecurePlatform login will either prompt the user for a keystore password, or it will get the password from another secure source and use it to unlock the KeyStore (see “Enabling platform single sign-on” on page 100).

Once the Java Keystore is unlocked, the Accounts API and the JAAS Login Modules can access secure data such as passwords stored in the Keystore. The Keystore can also be accessed programmatically by calling SecurePlatform.getKeyStore(). All Lotus Expeditor applications have access to the Java Keystore and can use it to store their own data.

Managing problem determination artifacts

Configure the Problem Determination Artifact Manager to ensure that the file system artifacts used for resolving errors are appropriately managed.

The Lotus Expeditor client runtime, while designed to run for extended periods without a restart, in some cases can experience unexpected error conditions. When these unexpected error conditions occur, in some cases Java heap dumps and core dumps can be generated to provide problem determination information for the service and support teams. Problem determination data can be critical in resolving these unexpected errors.

In other cases, problem determination may be more explicitly gathered by executing the IBM Support Assistant collection mechanisms built into the product. Both of these two cases produce file system artifacts in well-known locations on the client platform.

Over time, Java heap dumps and core dumps and ISA data collection jars can come to consume large amounts of disk space. To ensure that the file system space is managed appropriately, older files are deleted periodically. An aging algorithm is used to allow the files to remain for a configurable number of days to allow them to be referenced during a problem determination session, but once they have exceeded an age limit, they will be deleted. This aging period can be administratively configured and can also be administratively disabled. The default period is 5 days.

The Problem Determination Artifact Manager provides this capability for the Lotus Expeditor runtime client.

Changing the Problem Determination Artifact Manager default interval

You can change the Problem Determination Artifact Manager default interval to define when file system artifacts should be removed.

To change the Problem Determination Manager default interval, do one of the following:

• Using a text editor, open the <install dir>/rcp/plugin_customization.ini file and modify the com.ibm.rcp.core.pd.manager/interval property for the interval in days, and save the file.
  
  For example, com.ibm.rcp.core.pd.manager/interval=120.

• If you manage the client using the Device Manager server:
1. Launch the Device Manager server console to create a configuration job for the com.ibm.rcp.core.pd.manager/interval preference.

2. Schedule the job for the client device.

When the user launches the client platform, the device agent polls the server and receives the configuration job. The Eclipse preference is updated.

**Disabling the Problem Determination Artifact Manager**

This option lets you disable the Problem Determination Artifact Manager.

To disable the Problem Determination Artifact Manager, do one of the following:

- Using a text editor, open the `<install dir>/rcp/plugin_customization.ini` file and set the com.ibm.rcp.core.pd.manager/interval property to 0, and save the file.
  
  For example, com.ibm.rcp.core.pd.manager/interval=0.

- If you manage the client using the Device Manager server:
  
  1. Launch the Device Manager server console to create a configuration job for the com.ibm.rcp.core.pd.manager/interval preference setting it to 0.
  
  2. Schedule the job for the client device.

When the user launches the client platform, the device agent polls the server and receives the configuration job. The Eclipse preference is updated.

**Managing accounts**

You can store, access, and use properties that are required to make a connection to, and communicate with, a remote service.

The Accounts API provides a way to store, access, and use properties that are required to make a connection to, and communicate with, a remote service. Some examples of accounts include: An HTTP account which is used to connect to a web based service. This account will contain a URL for the location of the service, and a user name and password to logon to the service. An Instant Messaging account which is used by an IM client to connect to an IM server, such as IBM Lotus Sametime®. This account would include a server name, user name and password to connect to the IM server. The account could also be used to store user preferences such as the text people see when the user’s IM status is “Away”. So, an account can be used to store both connection properties, and properties or preferences specific to that connection. In the case of IM, the user could have multiple IM accounts, both for business and personal use. The “Away” status message can be stored in the account, so that it can be set to a different value for each one.

The Accounts API provides a way to get, add, update, remove, and listen for changes to, an account. Accounts can be obtained by unique id, name, or any other property, and a get account by server method provides for partial matches. The Accounts API also provides integration with the Java Authentication and Authorization Service (JAAS). (For more information about JAAS, refer to [http://java.sun.com/products/jaas/overview.htm](http://java.sun.com/products/jaas/overview.htm)) A properly constructed account can provide a login context service which will obtain and validate the credentials needed to connect to a remote service. Certain accounts will also obtain LTPA and session cookies. By using the Accounts integration with JAAS an application will have completely validated passwords and cookies needed to communicate with the remote service.

All accounts except for the Home Portal Account can be managed using the Accounts user interface. The Home Portal Account can be managed using the Home Portal Account user interface.

For more information, refer to [Accounts framework in Developing Applications for Lotus Expeditor](http://java.sun.com/products/jaas/overview.htm)
Creating accounts
An account holds a set of properties (name/value pairs) used by an application to connect to a remote service.

You can have an account to connect to your personal mail server using POP3 to get your email, an account to connect to a WebSphere Portal Server to access composite applications, or an LDAP account to do name look-up from the corporate directory. Some accounts may already exist on the client, either created by applications, or provisioned from Domino or Portal. Be careful when editing, removing, or creating new accounts, as this could cause some applications to no longer function. If you are not sure what to do, ask your administrator.

Advanced Properties for HTTP / HTTPS Accounts

The advanced properties for HTTP accounts are primarily used to specify additional properties needed to authenticate with either a Portal server directly or an HTTP/Portal server protected by Tivoli® Access Manager (TAM) or Site Minder. Knowing what to enter for these properties requires specific knowledge of the HTTP infrastructure of an organization, and should not be modified by users unless they are given specific instructions from their administrator. This section will provide enough information to use these properties, but assumes some knowledge of the authentication mechanism being used.

Authentication Types

HTTP Basic (default)
HTTP Basic is simple user name and password authentication. The authentication is done at the URL specified by the Server value and all other advanced properties are ignored. This is analogous to opening a Web browser to the URL and the browser popping up a dialog asking for a user name and password.

J2EE Form (advanced)
J2EE-FORM is a standard way of authenticating with a Web application server, such as WebSphere and Portal servers. A form is submitted to a URL that contains a user name and password. For WebSphere and Portal servers, an LTPA token is returned, which can be used for future authentication. J2EE-FORM uses the Authentication URL (auth URL) property to connect to a servlet. The auth URL is used in one of the following two ways:

- If the auth URL is a complete and valid URL (for example, http://myportalserver.com/wps/j_security_check), then the auth URL is used to locate the servlet.
- If the auth URL is a partial URL (for example, /wps/j_security_check), then it is appended to the root of the Server value.

For example, if the server value is http://myserver.com/mycontextroot and the auth URL is /wps/j_security_check, then the URL used will be http://myserver.com/wps/j_security_check.

Portal Form (advanced)
Portal Form is a more advanced version of J2EE-FORM. The auth URL is set to a slightly different location. This auth URL generates both an LTPA token for authentication and a JSession cookie for session data. Portal Form should be used to communicate with portlets, because J2EE-FORM does not provide the session cookie. Use J2EE-FORM to communicate with servlets hosted on Portal and WebSphere servers.

TAM Form and Site Minder Form (advanced)
TAM Form and Site Minder Form should be used when the HTTP resource is protected by one of these technologies. The HTTP resource should be configured to accept the authentication token provided by TAM or Site Minder. These auth types work in the same way as Portal form, except that a form is submitted that is specific to either TAM or Site Minder. The result of the authentication is a cookie that can be used to authenticate with any of the protected HTTP resources, including WebSphere and Portal servers.
User Name Token (advanced)

User Name Token is an authentication mechanism used by Web Services. This authentication type should only be used by child accounts that have selected the User name and password of an existing account. User name token should only be used if the Web Service does not accept any other form of authentication, since it requires sending the user name and password on every request. This authentication type is rarely used and should be avoided if possible.

TAM SPNEGO (advanced)

TAM SPNEGO is supported on Microsoft Windows operating systems only. It uses Active Directory to get a token based on the user operating system log-in. It then uses that token to authentication with TAM, rather than using a user name and password. Apart from that, the result of the authentication is the same as for TAM Form.

Authentication URL

The authentication URL is only used by J2EE-FORM and PORTAL Form because it needs to know where the authentication servlet is located. The value for this property can either be a complete URL or a path relative to the root of the server URL.

Home Portal URL

Some applications require this value to interact properly with a WebSphere Portal server. When accessing a Portal server via a web browser, by default the following URL would be used:

http://myportalserver.com/wps/myportal

The Home Portal URL in this case is the myportal part. Since this is configurable by the system administrator, the default value can be altered as needed in the advanced properties.

Managing Eclipse preferences

The platform and application plug-ins can use Eclipse preferences for storage of configurable information. Examples can include window settings, server locations, and synchronization settings.

Eclipse is an open source community committed to implementation of a universal development platform.

Eclipse preferences are managed in a tree format. The first nodes after the root are called scopes. The scopes each define a subtree. The default scopes provided by Eclipse are default, instance, configuration, and project. Under each of the scopes, the preferences tree is typically organized by plug-in identify. Associating preference keys and values with a particular plugin will help to eliminate naming collisions.

The product adds an additional scope called ManagedSettings. See "Configuring Managed Settings" on page 70 for more information.

Instance scope is stored per workspace, while the configuration scope is stored in the configuration directory. Because the platform install configuration sets the configuration directory to be a directory in the workspace, instance and configuration scopes are effectively equivalent.

You can search preferences by name, and the scopes can be searched in a particular order. A typical search order would be instance, configuration, then default scopes.

How are preferences set?

Preferences are set through the code, often as a result of interactions with preference pages. However, plug-ins can internally use preferences, and not surface those preferences through user interactions.
How are default preference values set?

The following list describes the several ways in which default preference values can be set for a particular plug-in. They are listed in the order in which they are processed. A default preference set at the lowest level (for example, Runtime defaults) is over-ridden by the same preference set at a higher level (for example, Product defaults).

Runtime defaults
These are specified by code, such as contained in a plug-in class or in a preference initializer supplied as an extension to the org.eclipse.core.runtime.preferences extension point

Bundle defaults
Default values specified in preferences.ini files in the plug-in.

Product defaults
Default values specified in the customization file supplied by the product.

Command line defaults
Default values specified in a plug-in customization file specified on the command line.

Once the developer has created the plug-in, at application assembly time, the assembler might choose to add preferences.ini files to the plug-in to override whatever the developer may have set.

Once plug-ins are deployed to a client system, it is strongly discouraged that the plug-in contents are updated. Therefore, after deployment of the plug-ins, the only applicable way to override preference default values is through the command line plug-in customization file. See “Updating the plugin_customization.ini file” on page 99.

For the client platform, the plugin_customization.ini is contained in the <installdir>/rcp directory.

After the client platform has been deployed, this file can be changed through the following methods:

- Manual update
- Code added to feature install handlers to modify the file. See “Using the global install handler” on page 151 for more information.
- Direct manipulation through properties files jobs supplied by the Device Manager server. See “Managing properties files” on page 203 for more information.

How are project, instance and configuration preference values set?

Settings in the project, instance and configuration scopes are typically made as a result of user interactions, such as through the preferences dialog. The plug-in code will need to use appropriate methods to search install and configuration scopes prior to using the default values.

Project preference values are typically not used by most platform and application components. Project level preferences are most often associated with code creation or application assembly tools that maintain a concept of projects, such as the Eclipse SDK.

Project, instance and configuration preference values can be queried after deployment using the preferences jobs supplied by the Device Manager server. See “Querying client runtime preferences” on page 202 for more information.

You can add, update, or remove project, instance, and configuration preference values using the following mechanisms:

- Feature install handlers during feature install. See “Using the global install handler” on page 151 for more information.
- Preferences jobs supplied by the device manager server. See “Managing preferences” on page 202 for more information.
• Policy settings supplied by the WebSphere Portal-administered client and server. See “Managing Rich Client Preferences using the Policy Type Editor for Eclipse Preferences” on page 234 for more information.

What about user-defined or custom scopes?

Applications can define additional custom scopes for storing preferences. The ManagedSettings scope defined by this product platform is an example of a custom scope. The preferences jobs and policy settings are used by the Eclipse preferences APIs to query and effect changes on the preferences values. The APIs will typically work for other scopes, unless the other scopes are providing additional restrictions. If an application is providing their own custom scope for preferences, the management operations supplied by the Device Manager server or the policy settings may not always perform successfully. Refer to the application for more information on how to manage values in its custom scope during deployment and post-deployment operations.

Managing OSGi preferences

The client platform includes an implementation of the OSGi Preferences service for use by OSGi-related applications.

The OSGi Preferences service implementation provided by Eclipse uses the standard Eclipse preferences service persistence mechanism for storage of preference values.

OSGi preferences are accessible only to the plug-ins that created the preference, and are associated with the bundle number of the plug-in. If the bundle number changes, the preferences will no longer be accessible. Operations such as starting the platform with the –clean option can cause bundles to be re-numbered. For this reason, OSGi preference usage is not encouraged.

How are preferences set?

Preferences are set directly through the code, often as a result of interactions with a user.

How are default preference values set?

Default preference values are always set by the service that is defining the preferences. The service may provide alternatives for setting the default values.

How can the preference values be changed after deployment?

You can query OSGi preference values after deployment using the preferences jobs supplied by the Device Manager server. Preference values for OSGi preferences are located at /configuration/org.eclipse.core.runtime.preferences.OSGiPreferences.xx where xx is the bundle number. See “Querying client runtime preferences” on page 202 for more information.

You can add, update, or remove project, instance and configuration preference values using the following mechanisms:

• Preferences jobs supplied by the Device Manager server. See “Managing preferences” on page 202.
• Policy settings supplied by the WebSphere Portal-administered client and server. See “Managing Rich Client Preferences using the Policy Type Editor for Eclipse Preferences” on page 234.

Since the storage mechanism contains some referential aspects, if a plug-in has not already created settings value, it is not recommended that you create a new OSGi preferences tree for the plug-in, but that you only modify values for the existing tree. Changes to OSGi Preference values may not be available until the next platform restart.
Managing UserAdmin settings

The client platform includes an implementation of the OSGi UserAdmin service. The UserAdmin service provides a mechanism for storing user definition, user credentials, user roles, and group information.

The UserAdmin service uses the OSGi Preferences mechanism for storage of this information. Since the OSGi Preferences mechanism uses the bundle number of the UserAdmin service to store user information, if the bundle number changes, the UserAdmin definitions will no longer be accessible. Operations such as starting the platform with the -clean option can cause bundles to be re-numbered.

The UserAdmin service is primary mechanism that can be used by the Web container for storing user and role related information.

How are user definitions created?

User definitions are created programatically leveraging the User Admin APIs.

How can user definitions be changed after deployment?

You can query UserAdmin settings after deployment using the preferences jobs supplied by the Device Manager server. UserAdmin settings are located at /configuration/org.eclipse.core.runtime.preferences.OSTiPreferences.xxx where xxx is the bundle number of the UserAdmin bundle (org.eclipse.equinox.useradmin). See "Querying client runtime preferences" on page 202 for more information.

Use the following mechanisms to add, update, or delete user settings:

- Preferences jobs supplied by the Device Manager server. See “Managing preferences” on page 202.
- Policy settings supplied by the WebSphere Portal-administered client and server. See “Managing Rich Client Preferences using the Policy Type Editor for Eclipse Preferences” on page 234. Since the storage mechanism contains some referential aspects, if the UserAdmin bundle has not already created any settings, it is not recommended that you create a new UserAdmin preferences tree for the plug-in, but that you only modify values for the existing tree. Changes to UserAdmin settings may not be available until the next platform restart.

Managing ConfigurationAdmin settings

The client platform includes an implementation of the OSGi ConfigurationAdmin service. The ConfigurationAdmin service provides a mechanism for configuration information for a service.

The ConfigurationAdmin service uses the OSGi Preferences mechanism for storage of this information. Because the OSGi Preferences mechanism uses the bundle number of the ConfigurationAdmin service to store user information, if the bundle number changes, the ConfigurationAdmin definitions will no longer be accessible. Operations such as starting the platform with the -clean option can cause bundles to be re-numbered.

How are user definitions created?

ConfigurationAdmin settings are primarily created through code. In addition, you can use the Admin Utility for OSGi to create user definitions.

How can ConfigurationAdmin settings be changed after deployment?

You can query ConfigurationAdmin settings after deployment using the preferences jobs supplied by the Device Manager server. Configuration settings are located at /configuration/
org.eclipse.core.runtime.preferences. OSGiPreferences.xx where xx is the bundle number of the ConfigurationAdmin bundle (com.ibm.osgi.service.cm). See “Querying client runtime preferences” on page 202 for more information.

Use the following mechanisms to add, update, or delete user settings
• Preferences jobs supplied by the Device Manager server. See “Managing preferences” on page 202.
• Policy settings supplied by the WebSphere Portal-administered client and server. See “Managing Rich Client Preferences using the Policy Type Editor for Eclipse Preferences” on page 234.

Since the storage mechanism contains some referential aspects, if the ConfigurationAdmin bundle has not already created any settings, it is not recommended that you create a new ConfigurationAdmin preferences tree for the plug-in, but that you only modify values for the existing tree. Changes to ConfigurationAdmin settings may not be available until the next platform restart.

---

**Managing life cycle**

This section describes the life cycle of the client platform.

**Understanding life cycle**

Use this section to get an understanding of life cycle.

**Plug-in life cycle**

Plug-ins present in the client platform have a defined life cycle and states. Plug-ins are present in one and only one state at a time. The defined states are as follows:

**Note:** Information in this section applies to using the Lotus Expeditor Client on a device, except where noted.

### Installed

The underlying framework has added the plug-in. Under normal operation, a plug-in registered by the Eclipse update components will enter a resolving process and transition to the resolved state. If a plug-in is present in the platform, but is in installed state, then either the plug-in has dependencies that have not been met, or another plug-in with an identical Bundle-SymbolicName has been selected by the framework. A plug-in in installed state cannot share packages, nor can it contribute extensions or extension points.

### Resolved

All of the prerequisites of the plug-in have been met. A plug-in may contribute packages, extensions, and extension points to the platform.

### Starting

The plug-in has been requested to start and the framework is in the process of starting the plug-in. Start operations should be fast, so any plug-in that remains in a starting state is indicative of a plug-in that is failing to use best practices in its BundleActivator class.

### Stopping

The plug-in has been requested to stop and the framework is in the process of stopping the plug-in. Stop operations should be fast, so any plug-in that remains in a stopping state is indicative of a plug-in that is failing to use best practices in its BundleActivator class.

**Active** The plug-in has been started.

**Uninstalled** The plug-in has been uninstalled and may not be used.
**Fragment life cycle**

Fragments are associated with a host plug-in. Since a fragment cannot be individually started and stopped, a fragment follows the life cycle associated with its host plug-in. Fragments are attached to their host plug-in during the resolution process.

**Platform Life Cycle**

When the platform is requested to start, the underlying framework will begin the initialization and startup process. To bootstrap the platform, a small number of plug-ins and fragments will be individually installed and started. This set of plug-ins is defined by the osgi.bundles property. The default value of this property is defined in the `<configdir>\config.ini` file.

The remainder of the plug-ins and fragments for the platform will be installed next. The set of plug-ins and fragments to install will be determined based upon the platform configuration. If the framework is using a managed only configuration, the set of plug-ins and fragments will be determined based upon the platform configuration defined in the platform.xml file residing in the `<configdir>`. If the framework is not using a managed only configuration, the set of plug-ins and fragments will be discovered by the Update Configurator component from the defined eclipse extension directories.

Once the installation of the plug-ins into the framework has completed, a resolver step will run and will attempt to resolve all of the plug-ins and fragments.

After the resolver step has completed, the framework will begin the process of starting the plug-ins. The framework uses settings known as Start Levels to organize the startup of the plug-ins. The framework exists at a specified level and the plug-ins are assigned to start at a specific level. The framework begins at level 0 with no plug-ins started. Next, the Framework will move to start level 1. All plug-ins assigned to start level 1 will then be started in some order. The framework will progress through each of the levels until it reaches the designated framework start level. The designated framework start level is determined by the osgi.startLevel property. The plug-ins that are started during the start level processing are those which have had a start level specifically assigned, and which have been marked to persistently start. A plug-in is marked for persistent start if it has been explicitly started by code that invokes the `start()` method on a Bundle object.

A specific set of plug-ins must always be started (active) for the platform to execute properly. The plug-ins required at startup are specified in the config.ini file in the configuration directory. The osgi.bundles property defines the set of plug-ins required at startup.

For any plug-in specified in this property, if it must start prior to the update configurator plug-in, then it must be able to resolve using only the plug-ins specified by this property. Any plug-in specified to start after the update configurator plug-in can use any other plug-ins installed by the update configurator or by the platform definition. Plug-ins in this property should generally be limited to those essential for startup.

After the framework has processed all of the required start levels, the framework will start running an application. An application as used here is a class referenced by an extension to the org.eclipse.core.runtime.applications extension point. Multiple plug-ins may define extensions to the org.eclipse.core.runtime.applications extension point. The actual application used at startup will be the application specified as a specific startup argument, or the application specified for a particular product.

This application defines the controlling characteristics of the platform, such as whether it will contain a workbench and provide a graphical user interface, or whether it will run to perform a specific task and then exit.

As the application runs, it will need the packages, extensions, extension points, and services contributed by the plug-ins in the platform. These plug-ins can automatically start when needed, based upon the
presence and value of the Eclipse-LazyStart attribute specified in the manifest.mf of the plug-in. If a plug-in is marked as Eclipse-LazyStart: true, then when a request to obtain a class from that plug-ins class loader is detected, if that plug-in is not already in the active state, then the framework will first attempt to start the plug-in to transition it from the resolved to active state. As part of the starting process, the BundleActivator start method will be invoked to perform any required initialization for the plug-in. If the plug-in is marked as Eclipse-LazyStart: false, then classes may be resolved from the plug-in without actually attempting to start the plug-in. Specifying Eclipse-LazyStart:false, or not specifying the Eclipse-LazyStart attribute at all, are equivalent in behavior.

Once the application is completed its task, the framework will begin the exit process. In reverse order of the start levels, the plug-ins in the platform will be called to stop, thus allowing the plug-ins to perform any needed cleanup operations to free resources.

This product has added some additional capabilities into the startup processing beyond the capabilities provided by the underlying Eclipse frameworks. First, the com.ibm.rcp.lifecycle.platform plug-in has been added to the osgi.bundles property. This plug-in processes configuration information to select a set of plug-ins, assign those plug-ins a start level, and mark those plug-ins for persistent start.

Secondly, the platform has defined the com.ibm.rcp.lifecycle.application.startBundles extension point, as well as APIs provided in the com.ibm.rcp.lifecycle.application package, to enable applications to allow assemblers and deployers the ability to specify that certain plug-ins must be started when an application starts. This extension point is defined to assist in the startup of plug-ins that provide services and that do not export any packages. Since they do not export packages, there will be no normal class references to these plug-ins, so the presence of the Eclipse-LazyStart attribute will have no effect. The set of plug-ins to start with an application is set uniquely per application. The APIs will attempt to start each of the plug-ins specified by the extension point. Using framework techniques, the APIs will attempt to start the plug-in, invoking the start() method on the plug-in only if the Eclipse-LazyStart: true attribute is not set. When the application completes its tasks, the APIs will attempt to stop each of the plug-ins that it has started.

When using the Lotus Expeditor Client on desktops, the default application provided by the client platform, identified as com.ibm.rcp.personality.framework.RCPApplication, provides support for personalities. Each personality can be thought of as a mini application. Correspondingly, there is a com.ibm.rcp.lifecycle.personality.startBundles extension point which provides the ability to start (and stop) plug-ins that are associated with a specific personality. All of the same characteristics for plug-in startup for a specific application discussed in the previous paragraph can be applied to plug-in startup for a specific personality. For more information, see Using Personalities in Developing Applications for Lotus Expeditor.

The Lotus Expeditor Client for Devices only supports the com.ibm.rcp.lifecycle.application.startBundles extension point. However, this extension point may also be used similarly to the personality.startBundles extension point to start plug-ins when the Lotus Expeditor Client environment starts on the device. See Understanding application lifecycle in Developing Applications for Lotus Expeditor for more details.

**Other notes on plug-in startup behavior**

The webcontainer provides an extension point com.ibm.pvc.webcontainer.application that allows local web and portlet plug-ins to identify themselves via their context root to the webcontainer without requiring the plug-in to be started. When the webcontainer receives an HTTP request, if the plug-in that provides a local web or portlet application for that context root is not yet started, it will be started by the webcontainer. This will only apply to applications that have a specific context root. Applications that contribute behavior to the / context root will not be automatically started. Applications in this category include those applications that use the OSGi HttpService to provide servlet applications.
What's the difference between a bundle and a plug-in?

The term plug-in originated with Eclipse. The term bundle originated from the OSGi Alliance specifications. Eclipse changed to use an OSGi framework as its underlying componentization model. For purposes of this startup section, there is no difference between a bundle and a plug-in.

How should I set the Eclipse-LazyStart attribute?

The recommendation is to set all plug-ins to have the Eclipse-LazyStart attribute with a value of true. This will allow the plug-ins to be started in an automatically (or lazy) manner, and not require them to be explicitly started and marked for persistent start.

Why should I avoid having my plug-ins marked for persistent start?

Plug-ins marked for persistent start will be started each and every time that the platform starts. This can result in slower startup performance. In addition, if the platform has no need of the contributions of the plug-in, the plug-in will still consume resources, contributing to poorer runtime performance or increased memory consumption.

What about compatibility with the previous releases?

In the previous releases, the product used the presence or non-presence of the Eclipse-LazyStart (then called Eclipse-AutoStart) attribute as a trigger for starting plug-ins. There are provisions in the com.ibm.rcp.lifecycle.platform configuration data and in the com.ibm.rcp.lifecycle.application.startBundles and com.ibm.rcp.lifecycle.personality.startBundles extension points to handle plug-ins that may have been created following the recommendations in the previous releases.

Contributing plug-ins to the personality life cycle

You can start a plug-in when a specific personality is started.

In the majority of cases, plug-ins that are exporting packages for use by other plug-ins, and contain the Eclipse-LazyStart: true attribute, are started automatically by the framework when there are requests for a class from that plug-in. In cases where a plug-in does not provide packages for use by other plug-ins, or the plug-in needs to start prior to later package references, then use this mechanism to start the plug-in with a personality.

Personalities allow you to customize the look and feel depending on the desired applications. Applications can have completely different requirements in terms of the plug-ins that are used, so when starting a personality, it is advantageous for resource reasons to start only what is needed. Because the Lotus Expeditor Client allows multiple personalities to run when using the default RCPApplication, you can control which plug-ins start based on a specific personality.

To contribute a plug-in when a specific personality is started:

1. Identify the target personality (com.ibm.rcp.platform.personality is the ID of the default personality used by the RCPApplication).
2. Add the plug-in to start either by adding the com.ibm.rcp.lifecycle.personality.startBundles to the service providing bundle or to another bundle independent of both the personality bundle or the service providing bundle.

The plug-in starts when the target personality is started.

A plug-in that needs to start DB2 Everyplace could define the following extension to start the com.ibm.db2e plug-in when the com.ibm.rcp.platform.personality starts:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<eclipse version="3.2">?
<plugin>
```
Contributing plug-ins to the application life cycle
You can start a plug-in when a specific Eclipse application is started.

In the majority of cases, plug-ins that are exporting packages for use by other plug-ins, and contain the Eclipse-LazyStart: true attribute, are started automatically by the framework when there are requests for a class from that plug-in. In cases where a plug-in does not provide packages for use by other plug-ins, or the plug-in needs to start prior to later package references, use the com.ibm.rcp.lifecycle.application.startBundles extension point to associate a plug-in with an Eclipse application. The Eclipse application must use the com.ibm.rcp.lifecycle.application.BundleControl API to process the extension point information and start the requested bundles. Refer to the com.ibm.rcp.lifecycle.application.BundleControl javadoc in Developing Applications for Lotus Expeditor for this class for more information and short sample code. The default application used by the client platform (com.ibm.rcp.personality.framework.RCPApplication) supports use of this extension point.

To contribute a plug-in to the application life cycle:
1. Identify the target application (com.ibm.rcp.personality.framework.RCPApplication is the ID of the default application).
2. Add an extension point to the bundle containing the application, add the extension point to the bundle containing the service, or add or modify some other bundle to add the extension point.

The plug-in starts when the target application is started.

A plug-in that needs to start DB2 Everyplace could define the following extension to start the com.ibm.db2e plug-in when the com.ibm.rcp.personality.framework.RCPApplication starts:

```xml
<extension
    point="com.ibm.rcp.lifecycle.application.startBundles">
    <application id="com.ibm.rcp.personality.framework.RCPApplication">
        <bundle id="com.ibm.db2e"/>
    </application>
</extension>
```

Platform users using the default com.ibm.rcp.personality.framework.RCPApplication can associate additional plug-ins to be started, as shown in the example above, and do not need to provide their own application. If the platform is not using the default application, plug-ins will need to be associated with the application used to start the platform. Bundles may be associated with more than one application. Since only one Eclipse application is used at startup, only the plug-ins associated with that application will be started.

Contributing plug-ins to the platform life cycle
You can start a plug-in when the platform is started.

Contributing plug-ins to the platform life cycle enables you to control which plug-ins are to start when the platform is started. In the majority of cases, plug-ins that are exporting packages for use by other plug-ins, and contain the Eclipse-LazyStart: true attribute, are started automatically by the framework when there are requests for a class from that plug-in. In cases where a plug-in does not provide packages
for use by other plug-ins, or the plug-in needs to start prior to later package references, then use this
time mechanism to start the plug-in with the platform. The primary reasons why plug-ins need to be started
with the platform is that they contain providers that are invoked by the base Java classes, and must be
registered prior to attempting to use the base Java class resources.

You have two methods of changing the platform startup configuration:

**Note:** It is possible to define a set of startup bundles that result in a platform that will not start. While
you may extend the set of bundles started at platform startup, it is not recommended to change
the start levels or set of default bundles.

- Deploy a new fragment to the com.ibm.rcp.lifecycle.platform host. If a fragment to the
  com.ibm.rcp.lifecycle.platform.plugin does not already exist, create a new fragment containing the file
customStartLevels.properties. The contents of the defaultStartLevels. properties from the
com.ibm.rcp.lifecycle.platform bundle are copied to the customStartLevels properties. See “Packaging
applications for deployment” on page 144 for more information on deploying fragments to the system.

  org.eclipse.equinox.log=1
  com.ibm.rcp.core.logger=3
  com.ibm.pvc.jndi.provider.java=5
  com.ibm.rcp.osgiagent.scheduler=10
  com.ibm.osgi.service.osgiagent.NativeInstallWatchDog=10
  com.ibm.rcp.net.http=10
  com.ibm.rcp.provisioning=10
  com.ibm.rcp.core.pd.manager=10

New customStartLevels.properties:

  org.eclipse.equinox.log=1
  com.ibm.rcp.core.logger=3
  com.ibm.pvc.jndi.provider.java=5
  com.ibm.rcp.osgiagent.scheduler=10
  com.ibm.osgi.service.osgiagent.NativeInstallWatchDog=10
  com.ibm.rcp.net.http=10
  com.ibm.rcp.provisioning=10
  com.ibm.rcp.core.pd.manager=10
  com.yourcompany.yourbundle=10

The new bundle is added to the list of bundles.

- Use system properties.
  The following system property would be expected to be used to make major changes to the default
start levels specified by the platform provider:
  - com.ibm.rcp.lifecycle.platform.defaultStartLevels - specifies the property file that contains the
    applicable start levels for the platform.

This property would be expected to be used to make major changes to the default start levels specified
by the platform provider.

The following properties would be expected to be used to add minor modifications to the default start
levels specified by the platform provider:

  - com.ibm.rcp.lifecycle.platform.startLevel.<startLevel>.include - applied after the start levels
    properties file has been processed. The <startLevel> is a specific start level to be modified from the
    value specified in the properties file. The value is a comma-separated list of bundles to be included
    at this particular start level. [services] and [legacy] may be used.

  - com.ibm.rcp.lifecycle.platform.startLevel.<startLevel>.exclude - applied after the start levels
    properties file has been processed. The <startLevel> is a specific start level to be modified from the
    value specified in the properties file. The value is a comma separated list of bundles to be excluded
    at this particular start level. [services] and [legacy] may be used.

  - com.ibm.rcp.lifecycle.platform.startLevel.<startLevel>.replace -- applied after the start levels
    properties file has been processed. The <startLevel> is a specific start level to be modified from the
value specified in the properties file. The value is a comma separated list of bundles to be started at this particular start level. [services] and [legacy] may be used. This value replaces the value contained in the properties file.

See "Updating the rcplauncher.properties file" on page 89 for more information on setting system properties.

If you need to make minor additions or changes to the default property set, such as when testing, then the system properties may be sufficient. If you need to make more significant changes, or need to deploy changes to many systems, a new fragment may be a better option.

The plug-in starts when the platform is started.
Assembling and deploying

This section describes how to assemble and deploy Lotus Expeditor Client.

Extending the platform

This section describes how to extend the platform.

Specifying platform branding and theme

The Lotus Expeditor Client Workbench leverages Eclipse products and Eclipse themes for branding. You should create your own branding plug-in with your product information instead of changing the one shipped with the Lotus Expeditor Client. Create your own branding feature and deploy it as part of your initial install.

A developer follows these steps to change the Workbench branding:

1. Create a new branding plug-in (for example, com.myproduct.branding).
2. Create a plugin.xml file with Eclipse Products extension in the com.myproduct.branding plug-in.
3. Include the preferenceCustomization property in the Products extension to specify the preference customization file to use.
4. Include an Eclipse Themes extension in the plugin.xml file in the com.myproduct.branding plug-in.
5. Include a preference customization file that is specified in the Eclipse Products extension. This is typically the plugin_customization.ini file in the com.myproduct.branding plug-in. This should contain the preference for setting the current Theme.
6. Include the splash screen to be used for the product in the plug-in.
7. Create a feature for the above plug-in.
8. Use the global install handler to specify the branding feature’s product attribute in the rcplaunch.properties file so that the correct product and application will be used on launch. See "Using the global install handler" on page 151 for details.

For example, product=com.ibm.myproduct.branding.<MyProduct> where <MyProduct> is the ID specified in the Products Extension in step 2.

9. Use the global install handler to set the branding feature’s osgi.splashPath Java system property (-Dosgi.splashPath=) to change the splash screen in the rcplaunch.properties file. The syntax should be URL syntax. For example, platform\:/base/..\shared\eclipse\plugins\com.myproduct.branding.

10. Deploy the branding feature as part of the initial install by updating the provisioning manifest.

Providing your own product branding

Product branding is specified in the Eclipse product extension point.

The following is the code example for default branding used by the Lotus Expeditor Client:

```xml
<extension
   id="DefaultProduct"
   point="org.eclipse.core.runtime.products">
   <product
      name="BigDecimal"
      application="com.ibm.rcp.personality.framework.RCPApplication"
      description="BigDecimal">
      <property name="appName" value="DefaultPlatform" />
      <property name="aboutText" value="BigDecimal" />
      <property name="windowImages" value="icons/16.gif,images/32.gif" />
      <property name="aboutImage" value="icons/productAbout.bmp" />
```

© Copyright IBM Corp. 2005, 2008
To deploy your own branding, you must create a new branding plug-in and feature and deploy them as part of the initial installation by updating the provisioning manifest. The new branding feature must use the global install-handler to modify the product attribute in the rcplauncher.properties file.

Use the above snippet as an example for providing your branding. Replace com.ibm.rcp.platform.personality.branding in the above snippet with the name of your branding plug-in.

**Specifying the Window title**

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 specify the name attribute of the product extension in the plugin.xml file of your branding plug-in.

Code example of the product extension in the plugin.xml file:

```xml
<extension
  id="MyProduct"
  point="org.eclipse.core.runtime.products">
  <product name="%product.name"
    application="com.ibm.eswe.workbench.WctWorkbenchApplication"  
    description="%product.description">
  </product>
</extension>
```

Provide the actual value of the product name in the plugin.properties file of your branding plug-in.

Code example from plugin.properties:

```
# Product Name
product.name=MyProduct
```

Code example from plugin_<locale>.properties:

```
# Locale Specific Comment
product.name=<Locale-specific string>
```

**Specifying the Window image**

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.

Code example:

```xml
<extension
  id="MyProduct"
  point="org.eclipse.core.runtime.products">
  <product name="%product.name"
    application="com.ibm.eswe.workbench.WctWorkbenchApplication"
    description="%product.description">
  </product>
</extension>
```
Specifying the About dialog image

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

The image shown in the About dialog must be a file with a .bmp extension. A full-sized product image (no larger than 500 x 330 pixels) is shown without the "aboutText" text. A half-sized product image (no larger than 250 x 330 pixels) is shown with the "aboutText" text beside it.

Code example:
```
<extension id="MyProduct" point="org.eclipse.core.runtime.products">
  <product name="%product.name" application="com.ibm.eswe.workbench.WctWorkbenchApplication"
    description="%product.description">
    <property name="windowImages" value="platform:/plugin/com.myco.example/exampleimages/16.gif,
      platform:/plugin/com.myco.example/exampleimages/32.gif" />
    <property name="aboutImage" value="platform:/plugin/com.myco.example/exampleimages/productAbout.bmp"/>
    <property name="aboutText" value="%productAboutText"/>
  </product>
</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 com.myco.example.nl_<version_number>_exampleimages\fr directory.

If the system does not find a About dialog image for your locale, then the system selects the default image from the com.myco.example_<version_number>_exampleimages directory.

Specifying the About dialog text

Specify the text that is displayed next to the image in the About dialog in your branding plug-in's plugin.properties file. The plugin.properties file is locale-specific.

Code example:
```
<extension id="MyProduct" point="org.eclipse.core.runtime.products">
  <product name="%product.name" application="com.ibm.eswe.workbench.WctWorkbenchApplication"
    description="%product.description">
    <property name="windowImages" value="platform:/plugin/com.myco.example/exampleimages/16.gif,
      platform:/plugin/com.myco.example/exampleimages/32.gif" />
    <property name="aboutImage" value="platform:/plugin/com.myco.example/exampleimages/productAbout.bmp"/>
  </product>
</extension>
```

From plugin.properties:
```
# About Text
productAboutText="Critical Product Information"
```
You can have different About dialog text for each locale that the product supports. For example, from plugin_<local>.properties in the com.myco.mytheme.nl<locale group number> fragment:

```xml
# Locale Specific Comment
productAboutText=Locale Specific String
```

### Specifying 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 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 use your own splash screen, include the splash image in your branding plug-in. In your branding feature, use the global install handler to set the osgi.splashPath Java system property (-Dosgi.splashPath=). The syntax should be URL syntax. For example, -Dosgi.splashPath=platform:/base/../shared/eclipse/plugins/com.ibm.sample.screen.

The splash screen is used to display status during workspace creation and also when processing provisioning operations. If you change the splash screen from the default image, then you will also need to re-configure the foreground text color and coordinates of the status message and progress bar rectangles.

To use the splash progress monitor, the foreground message color, message area, and progress bar area must be configured. The message area and progress bar area are defined as "x,y,length,width" relative to the splash image. Color is defined as a 6-character hex String - first 2 represent red, second 2 green, third 2 blue. The use of the splash progress monitor will perform a look-up in the following sequence to identify the message area and progress bar locations. Message and progress rectangles must be set or the settings from that group would not be used. In other words, if a product definition is found that only had a progress rectangle bar definition and not a message rectangle bar, that option is skipped and the next option is processed.

1. As properties in the product definition being launched. This depends on Platform.getProduct() returning a product definition.

```xml
<extension
    id="BaseProduct"
    point="org.eclipse.core.runtime.products">
    <product
        name="BaseProduct" application="com.ibm.rcp.personality.framework.RCPApplication"
        description="%product.description">
        <property name="appName" value="DefaultPlatform" />
        <property name="aboutText" value="%productAboutText" />
        <property name="windowImages" value="icons/16.gif,icons/32.gif" />
        <property name="aboutImage" value="icons/productAbout.bmp" />
        <property name="preferenceCustomization" value="platform:/plugin/com.ibm.rcp.platform.personality.branding/plugin_customization.ini" />
        <property name="startupMessageRect" value="0, 0, 400, 20" />
        <property name="startupProgressRect" value="0, 400, 400, 20" />
        <property name="startupForegroundColor" value="ABCDEF" />
    </product>
</extension>
```

While this can be used to define the properties, the product definition is not available during workspace creation.
2. As preferences associated with the product specified during launch. This looks through the command line args to find -product and the associated name. Uses this as the nodename to locate preferences. Would allow products to define their own locations for message and progress bar. For example, add these to the plugin_customization.ini. If starting with com.ibm.rcp.platform.personality.branding.DefaultProduct, these options would be used:

com.ibm.rcp.platform.personality.branding.DefaultProduct/startupForegroundColor=00FF00
com.ibm.rcp.platform.personality.branding.DefaultProduct/startupProgressRect=6,6,410,20
com.ibm.rcp.platform.personality.branding.DefaultProduct/startupMessageRect=6,210,487,10

3. As preferences associated with com.ibm.rcp.provisioning. Similar to 2, exception that instead of a product name, com.ibm.rcp.provisioning is used as the node name

4. As system properties.

```bash
rcplauncher -vmargs -DstartupForegroundColor=CAFE00 -DstartupProgressRect=4,5,6,7 -DstartupMessageRect=1,2,3,4
```

System properties can be specified in the rcpinstall.properties, as system properties in config commands in the rcplauncher.properties file.

**Specifying your own help home page**

Help Contents for the product can be accessed by clicking Help → Help Contents. To change the default help home page to one for your own product, provide the new help page as a fragment to org.eclipse.help.base. You must use a different name from what is currently used (for example, help_home.html and rcp_help_home.html are already used). The default help fragment provided by the Lotus Expeditor Client is com.ibm.rcp.platform.personality.branding.help. It is recommended that the you use this as an example to create your own help fragment. Making changes to the one provided by the Lotus Expeditor Client and using it is not recommended.

The product branding plug-in should contain the help page preference in the product plugin_customization.ini, such as:

```
```

**Specifying the theme**

The overall look and feel of the client is driven by a combination of Eclipse themes extensions and cascading style sheet (CSS) content.

Elements specific to product branding such as the color or visibility of the banner bar are driven by dedicated data key/value elements of an Eclipse theme extension. The look and feel for items such as the launch button, toolbars, and presentation factory is driven by the CSS content.

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 applications contributed through extension points or through composite application definitions.

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:

- Banner images
- Banner background color
- Banner application title (font, position, color, and text)

**Specifying the banner image**

You can specify the banner images in your branding plug-in’s plugin.properties file. The following is an Eclipse theme example:
<extension
    point="org.eclipse.ui.themes">
    <theme id="MyProductGreenTheme">
    <description>
        This is the My Product Green Theme
    </description>
    <data
        name="com.ibm.rcp.platform.BRANDING_BAR_IMAGE_BACKGROUN"n
        value="platform:/plugin/com.myco.example/exampleicons/strip.e.gif">
    </data>

    <data
        name="com.ibm.rcp.platform.BRANDING_BAR_IMAGE_LEFT"
        value="platform:/plugin/com.myco.example/exampleicons/brand_bar_left.gif">
    </data>

    <data
        name="com.ibm.rcp.platform.BRANDING_BAR_IMAGE_RIGHT"
        value="platform:/plugin/com.myco.example/exampleicons/brand_bar_right.gif">
    </data>

    <data
        name="com.ibm.rcp.platform.BRANDING_BAR_VISIBLE"
        value="true">
    </data>

    Specifying 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 specify the
    background to display your own image.

    A background image smaller than 500 x 330 pixels will be tiled in the
    x and y directions. A background image 500 x 330 pixels or larger will
    be centered.

    Code example for the default background image used by the Lotus
    Expeditor Client:

    <extension
        point="org.eclipse.ui.themes">
        <theme id="MyProductGreenTheme">
        <description>
            This is the My Product Green Theme
        </description>
        <data name="WORKBENCH_BG_IMAGE"
            value="platform:/plugin/com.ibm.rcp.platform.personality.branding
            /icons/default_background.gif"/>
    </extension>

    Use the above code example as an example for specifying your own
    background image. Replace com.ibm.rcp.platform.personality.branding
    with the name of your branding plug-in.

    You can also change the image path and file name by specifying the
    defaultBackgroundImage property in the plugin_customization.ini file in
    the plugin.xml file of your branding plug-in.

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

    Specifying the banner color

    Lotus Expeditor: Assembling and Deploying Lotus Expeditor Applications
You can specify the banner color in your branding plug-in’s plugin.properties file. The following is an Eclipse theme example:

```xml
<extension
    point="org.eclipse.ui.themes">
    <theme id="MyProductGreenTheme">
        <description>
            This is the My Product Green Theme
        </description>
        ...
        <colorDefinition
            label="%BrandingBarForegroundColor"
            value="255,255,255"
            id="com.ibm.rcp.platform.BRANDING_BAR_FOREGROUND_COLOR">
            <description>
            </description>
        </colorDefinition>
        <colorDefinition
            label="%BrandingBarBackgroundColor"
            value="0,0,0"
            id="com.ibm.rcp.platform.BRANDING_BAR_BACKGROUND_COLOR">
            <description>
            </description>
        </colorDefinition>
        <colorDefinition
            label="%BrandingBarBackgroundColorBegin"
            value="56,56,56"
            id="com.ibm.rcp.platform.BRANDING_BAR_BACKGROUND_COLOR_BEGIN">
            <description>
            </description>
        </colorDefinition>
        <colorDefinition
            label="%BrandingBarBackgroundColorEnd"
            value="80,80,80"
            id="com.ibm.rcp.platform.BRANDING_BAR_BACKGROUND_COLOR_END">
            <description>
            </description>
        </colorDefinition>
        ...
</theme>
</extension>
```

**Specifying the banner application title font**

You can specify the banner color in your branding plug-in’s plugin.properties file. The following is an Eclipse theme example:

```xml
<extension
    point="org.eclipse.ui.themes">
    <theme id="MyProductGreenTheme">
        <description>
            This is the My Product Green Theme
        </description>
        ...
        <data
            name="com.ibm.rcp.platform.BRANDING_BAR_FONT_POINT_SIZE"
            value="14">
        </data>
    </theme>
</extension>
```

**Specifying the status line**

You can specify if your product should have a Status Line or not in your branding plug-ins plugin.properties file. The following is an Eclipse theme example:
Specifying a theme using CSS content

To contribute a theme using CSS content, you need to first extend the com.ibm.rcp.ui.themes extension point. Themes extensions specify CSS content, a human-readable name for the theme, and an optional preview image for the theme. Following is an example of a theme extension. Refer to the Themes schema in Developing Applications for Lotus Expeditor.

```xml
<extension point="com.ibm.rcp.ui.themes">
  <theme id="com.ibm.rcp.ui.themes.mytheme"
    label="MYTEST Theme"
    css="themes/global.css"
    previewImage="themes/mytesttheme_preview.png">
  </theme>
</extension>
```

The current theme is specified using an Eclipse preference. The preference ID used to specify the theme is WED_THEME_PREFERENCE in the com.ibm.rcp.ui namespace. This preference should be specified in plugin_customization.ini of your branding plugin. Following is an example of how to specify the value of this preference:

```ini
com.ibm.rcp.ui/WED_THEME_PREFERENCE= com.ibm.rcp.ui.themes.mytheme
```

Next, create a CSS file and add your new styles to it. Note that you will need to import the platform CSS file. The syntax for importing your CSS file is as follows:

```css
@import url("platform:/plugin/com.ibm.rcp.ui.css/themes/wed61/global.css");
```

Distributing branding updates

To deploy your own branding, create a new branding plug-in and feature and deploy it as part of the initial installation by updating the provisioning manifest or using any of the provision capabilities.

The product property in the rcpinstall.properties file must be updated to refer to product ID defined by your new branding plugin. See “Updating the rcpinstall.properties file” on page 93 for more information.

Configuring the Portlet Container branding

You can modify the look and feel of portlets. Use the instructions in this section to modify the portlet user interface elements used by the Portlet Container.

The Portlet Container branding extension applies to all portlets deployed to the IBM Lotus Expeditor runtime.

These instructions include the creation and deployment of a branding Web application. This Web application will include the portlet user interface elements including the HTML style sheet, the icons, and the screen colors.

To modify the look and feel of portlets, follow these steps:

1. Create a Client Services Web project. Refer to "Developing Web applications" in Developing Applications for Lotus Expeditor for instructions on how to create a Web application.
2. Select **Import → General → File System** to add the branding resources (stylesheet, and icons, for example) to the Web application.

3. Open the plugin.xml descriptor and add a `com.ibm.rcp.portletcontainer.branding.theme` extension contribution.

### 6.1.2 The following is an example of a branding extension for Expeditor 6.1.2:

```xml
<extension point="com.ibm.rcp.portletcontainer.branding.theme">
  <theme styleSheet="/theme/Styles.css" titleBarColor="#5495D5" contentColor="#FFFFFF"/>
</extension>
```

### 6.1.1 The following is an example of a branding extension for Expeditor 6.1.1:

```xml
<extension point="com.ibm.rcp.portletcontainer.branding.theme">
  <theme styleSheet="/theme/Styles.css" titleBarColor="#5495D5" contentColor="#FFFFFF"/>
</extension>
```

**Note:** Administrators should modify the resource paths to match how the resources are packaged in the Web application.

4. Click the **Save** button to save all changes made to the plugin.xml descriptor. The branding Web application is now ready to be distributed to the IBM Lotus Expeditor runtime.

5. See "Packaging applications for deployment" on page 144 for instructions on how to distribute the branding Web application.

### Configuring enterprise definitions (JNDI)

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

#### Lotus Expeditor Client JNDI overview

The Lotus Expeditor Client runtime provides a simple Java object JNDI registry to support the enterprise object definition needs of web applications, EJB applications, and messaging applications.

The Lotus Expeditor Client 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 name spaces, rather it provides a simple hierarchical name space 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 Lotus Expeditor Client JNDI provider when a CompoundName object is needed.
The Lotus Expeditor Client JNDI provider can be directly accessed via the provider’s `InitialContextFactory` class as shown below:

```java
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 Lotus Expeditor Client platform so even if no provider is specified it will still use the above `InitialContextFactory` to generate the `InitialContext` object.

The Lotus Expeditor Client 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 programatically register the objects that it needs each time the platform starts, the Lotus Expeditor 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-LazyStart: true` entry.

The Lotus Expeditor Toolkit leverages this declarative JNDI capability to automatically generate the required `plugin.xml`, and `Manifest.mf` entries for EJBS so that the Lotus Expeditor Client 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:

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

Note that a required component of the `com.ibm.pvc.jndi.provider.java.binding` extension point is the `objectFactory-id`. The Lotus Expeditor Client provides the following `ObjectFactory` implementations:
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. The TxnDataSourceObjectFactory provides the ability to create and bind into JNDI transaction capable DataSource objects that are required by the embedded transaction container. The WSOBJECTFactory allows the Web Services stub creator to provide information about the stub so it can be identified on a JNDI lookup call. The ManagedDSObjectFactory provides support for managed JDBC DataSources, while the JMSConnectionFactory, JMSTopicFactory and JMSQueueFactory provide JMS support for the micro broker platform messaging components.

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.bindin in their plugin.xml files. Some object factories may also provide another extension point that needs to be implemented as well, such as the GenericObjectFactory.

The ManagedDSObjectFactory provides an Eclipse extension point (described using a schema definition file) that will allow for the description of managed Java JDBC DataSource objects to bound into JNDI on a client JNDI lookup. The ManagedDSObjectFactory is responsible for processing declarative definitions of the com.ibm.rcp.database.core.datasource extension point and creating the JDBC DataSource object to be managed. For more information on the usage of this objectFactory, see "Defining managed data sources" on page 128, the schema definition for the the com.ibm.rcp.database.core.datasource extension point, and the Database Lifecycle Management section in Developing Applications for Lotus Expeditor.

The JMSConnectionFactory, JMSTopicFactory, and JMSQueueFactory object factories provide Eclipse extension points (described using a schema definition file) that will allow for the description of Java JMS objects to bound into JNDI upon a client JNDI lookup. These objectFactories are responsible for processing declarative definitions of the com.ibm.msg.client.jmsconnectionfactory, com.ibm.msg.client.jmsopic and com.ibm.msg.client.jmsqueue extension points and creating the JMS objects to support enterprise messaging in IBM Lotus Expeditor. For more information on the usage of these extension points, see the Declaratively registering JMS administered objects section in Developing Applications for Lotus Expeditor.

EJBObjectFactory:

The EJBObjectFactory is responsible for managing the life cycle of EJ Bs in the Lotus Expeditor Client runtime.

Upon a client lookup of an EJB, the JNDI provider will use the EJBObjectFactory to locate the EJB and start it, which in turn will cause the EJB to register with the EJB container and be bound into JNDI. The Lotus Expeditor Toolkit will automatically generate the appropriate plugin.xml entries for EJ Bs as a part of the deployment process, and will also add the Eclipse-LazyStart: true entry into the Manifest file to ensure that the EJB will not be started automatically upon platform start, rather it will be started by the EJBObjectFactory upon JNDI client lookup.

GenericObjectFactory:
This factory provides an Eclipse extension point (described via a schema definition file) that will allow for the description of Java objects to bound into JNDI upon a client JNDI lookup.

An example of the use of the new extension point to instantiate a DB2 Everyplace DataSource object is as follows:

```xml
<extension point="com.ibm.pvc.jndi.provider.java.genericobject">
  <object
    jndi-name="java:comp/env/jdbc/dsname"
    class="com.ibm.db2e.jdbc.DB2eDataSource"
    <method name="setUrl">
      <method-parameter
        type="String"
        value="jdbc:db2e:oedb">
      </method-parameter>
    </method>
  </object>
</extension>
```

The `com.ibm.pvc.jndi.provider.java.genericobject` extension point definition allows for:
- the `jndi-name` of the object
- one class name to be specified per object entry,
- a list of parameters including type (supported types listed below) and value to be used to create the constructor call to be executed to create this object,
- a list of methods to be called against this object including parameters with type (supported types listed below) and value to be used in the method calls.

**Note:** The parameter values can include Java system properties that will be resolved when the object is instantiated. Specify a Java system property using the syntax `${propertyname}`. If the property does not exist, the property will be replaced by an empty string. Special properties `_workspace` and `_configuration` have been defined and refer to the current workspace or configuration areas, based upon the method calls to `org.eclipse.core.runtime.Platform.getInstanceLocation()` and `getConfigurationLocation()`. The following is an example of the use of this property replacement:

```xml
<method-parameter
  type="String"
  value="jdbc:db2e:${_workspace}/oedb">
</method-parameter>
```

The `jndi-name` and `class` attributes are the only required attributes.

The list of valid types for the parameters is as follows:
- **Objects:** Boolean, String, Integer, Short, Long, Float, Double
- **Primitives:** boolean, int, short, long, float, double

**TxnDataSourceObjectFactory:**

The `TxnDataSourceObjectFactory` declarative JNDI component allows for this object transformation to be declaratively described via eclipse extension points.

To create transaction capable data sources which are required by the embedded transaction container of the Lotus Expeditor Client runtime, standard DataSource objects, such as the `com.ibm.db2e.jdbc.DB2eDataSource` are passed to the `com.ibm.pvc.txncontainer.TxnDataSourceFactory.create()` method and a suitable DataSource for use in embedded transactions is returned. `TxnDataSourceObjectFactory` leverages the same `com.ibm.pvc.jndi.provider.java.genericobject` extension point as the `com.ibm.pvc.jndi.provider.java.genericobjectfactory`, so the definition of the data source is the same, but in the definition of the binding itself, the `TxnDataSourceObjectFactory` is used.
Note: Since the point of this ObjectFactory is to create transaction capable DataSources, the value for the class element of the genericobject extension point must implement javax.sql.DataSource. If the class provided does not, the Factory will throw an InvalidObjectException and the JNDI object being located will not be found.

The following examples shows how to create a DB2eDataSource that can be used by the embedded transaction container.

```xml
<extension point="com.ibm.pvc.jndi.provider.java.binding">
  <binding
    jndi-name="java:comp/env/jdbc/txndsname"
    objectFactory-id="com.ibm.pvc.txncontainer.TxnDataSourceObjectFactory">
  </binding>
</extension>

<extension point="com.ibm.pvc.jndi.provider.java.genericobject">
  <object
    jndi-name="java:comp/env/jdbc/txndsname"
    class="com.ibm.db2e.jdbc.DB2eDataSource">
    <method name="setUrl">
      <method-parameter
        type="String"
        value="jdbc:db2e:oedb"/>
    </method>
  </object>
</extension>
```

**WSObjectFactory:**

The WSObjectFactory is responsible for providing a pre-initialized instance of Web services client stub for Apache Axis in the IBM Lotus Expeditor runtime.

Upon a client lookup of the client stub using JNDI, the JNDI provider will use the WSObjectFactory to locate the Web service stub based on the extension definition, which in turn will cause the stub to initialize and be bound into JNDI.

The IBM Lotus Expeditor Toolkit automatically generates the appropriate plugin.xml entries for the stub as a part of the development process. The stub developer or the deployer will be required to fill in appropriate values.

The following example shows how to register the service interface with the Wsfactoryobject:

```xml
<extension point="com.ibm.pvc.jndi.provider.java.binding">
  <binding
    jndi-name="com.ibm.test.EchoService"
    objectFactory-id="com.ibm.rcp.ws.objectfactory.WSObjectFactory">
  </binding>
</extension>

<extension point="com.ibm.rcp.ws.objectfactory.WSfactoryobject">
  <WSobject
    account-key="username.us.ibm.com"
    jndi-name="com.ibm.test.EchoService"/>
</extension>
```

Where: account-key is the name of the account, class is the class name of the service interface, and jndi-name is the name client will use to lookup the stub instance.

**Extending declarative JNDI**

The declarative JNDI solution leverages the Eclipse extension point registry to provide a means to declare a list of objects that should be bound in JNDI if and when a client application attempts to locate them.
To provide an extensible mechanism for this the declarative JNDI component makes use of the JNDI
ObjectFactory interface as a way of abstracting the JNDI provider from the specific implementation code
needed to instantiate different objects to be bound in JNDI. The javax.naming.spi.ObjectFactory
interface is very simple and contains only one method getobjectInstance().

getobjectInstance() is called by the lookup() method of our JNDI provider implementation if it is
unable to locate the object requested in it’s current registry of JNDI objects. The JNDI provider will first
read the list of object factories from an Eclipse extension point. It will then read the list of descriptions of
objects to be bound into JNDI. With this information, the JNDI provider will determine if the name that
is attempting to be located is in the list of names to be lazily bound, if it is, it will instantiate the
ObjectFactory based on the id provided in the extension point registry for that name and call
getobjectInstance() with a null object, the current name and context. The returned object from this
method invocation will then be bound into the JNDI object registry and returned from the lookup
method of the JNDI provider.

In the Lotus Expeditor Client runtime Object factories are registered via the Eclipse extension point
registry which allows for any number of ObjectFactory implementations to be registered and available to
the JNDI Provider.

This extension point is com.ibm.pvc.jndi.provider.java.objectfactory and an example of the usage of
this extension point would be similar to the following:

<extension
  point="com.ibm.pvc.jndi.provider.java.objectfactory">
</extension>

Life cycle management of JNDI registry:  It is required that JNDI be notified when a lazily bound object
is removed from the environment so that it can be sure to unbind the object from the JNDI registry. This
requirement will be met by registering a listener with the extension registry which will notify JNDI when
new objects are registered and also when currently registered objects are unregistered. When this
notification occurs the JNDI provider will add the new object or remove the existing object from its
registry.

It is the responsibility of the JNDI provider to manage the life cycle of the binding, not of the instantiated
object. The ObjectFactory implementation is responsible for managing the life cycle of the created object
by registering any appropriate listeners, and managing the cleanup of any associated resources if the
contributing bundle artifact is de-activated. It may do so by registering its own Extension Registry
listener (org.eclipse.core.runtime.IRegistryChangeListener) for the exposed extension point.

Defining managed data sources

You can declare, migrate, and populate databases that your organization’s Lotus Expeditor Client
applications access using the Database Life Cycle Management component.

The Database Life Cycle Management component allows Lotus Expeditor Client applications to
interoperate with a relational database. As administrator, you can specify the required database structure
(schema) and then populate the database before client access. You can migrate the database as well.
Database interaction is broken down into three type of tasks: create, migrate, and populate. Create tasks
run first and usually contain instructions that are responsible for creating tables in a database. Migrate
tasks run after the create tasks and are responsible for migrating any data that currently exists in the
database. Populate tasks run last and usually contain instructions that are responsible for populating the
database with new data. If one of these tasks fails, an error message is logged and the data source is not
accessible.

An advantage of using the schemas is that they are versioned. If multiple schema versions are deployed
on the system, the schemas are applied sequentially based on the version. For example, if you deploy
both a 1.0 version and 2.0 version of the schema, the 1.0 schema tasks run first, followed by the 2.0
schema tasks. Since the schemas are applied sequentially, successive versions of schemas should contain
only the deltas from the previous versions. For example, if schema version 1.0 creates table TABLE1 in
the database, then schema version 2.0 does not need to create table TABLE1 because it was already
created by schema version 1.0. Prior schema versions should not be removed from the platform, since
they might need to be applied by new users on the platform.

Consider the following scenario. An application is initially deployed and uses schema version 1.0. Since
this is a new application and a new database definition, users using this application will apply schema
1.0 to the database. Later, an application update is required and schema version 2.0 is deployed. Existing
users will only apply schema version 2.0 to the database, since they have already applied version 1.0.
Since version 1.0 is already deployed, the instructions in schema version 2.0 do not need to repeat version
1.0 instructions. A new user, however, will apply version 1.0 first (since the database has not been
created) and then version 2.0.

With the Lotus Expeditor Client, a database can be a specific instance of any type of SQL database,
tabular data source, spreadsheet, or flat file that can provide access to its content using a JDBC-compliant
driver.

The steps involved in defining data sources involves using the Lotus Expeditor Client Toolkit with
extension point schemas. Extensions are the key mechanism that a plug-in uses to add new features to
the platform. Extensions cannot be arbitrarily created. They are declared using a clear specification
defined by an extension point. Each extension must conform to the specification of the extension point
that it is extending. Each extension point defines attributes and expected values that must be declared by
an extension. This information is maintained in the platform plug-in registry. Extension point providers
query these values from the registry, so it’s important to ensure that your plug-in has provided the
expected information.

**Table 15. Database Life Cycle Management Extension Points**

<table>
<thead>
<tr>
<th>Extension point</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.rcp.database.core.provider</td>
<td>This extension point allows for the definition of new database providers. If you want to use another database besides the platform’s default Derby database, you must implement this extension. Once the extension is implemented, users of the platform may access the provided database via the database provider ID.</td>
</tr>
<tr>
<td>com.ibm.rcp.database.core.schema</td>
<td>This extension point allows for the definition of schemas. A schema is a set associated create, populate and migrate tasks. Schema definitions do not necessarily correspond to a specific database vendor’s schema, but form the logical unit of a database that undergoes life cycle events. Each schema is versioned independently, and may contain its own set of tasks to maintain its life cycle.</td>
</tr>
<tr>
<td>com.ibm.rcp.database.core.datasource</td>
<td>This extension point is used to declare a managed data source to the database infrastructure. A managed data source allows your company’s custom applications to connect a database provider with associated schemas to manage life cycle of the data source. If declared, the data source will be created and associated with a ManagedDatasourceObjectFactory. From this, an instantiated version of the DataSource object can be retrieved from JNDI, via a context.lookup().</td>
</tr>
<tr>
<td>com.ibm.rcp.database.core.listener</td>
<td>This extension point allows your custom applications to define listeners to hook into data sources. Listeners are useful when you want to know the status of operations being applied to their data source. For example, you may want to inform the user of some progress if extensive schema processing on a data source is about to occur.</td>
</tr>
</tbody>
</table>
Choosing between datasource types

The Managed Datasource components provides a default managed datasource definition (JNDI name jdbc/DerbyDS) that can be used by all applications. If you want to use the Managed Data Source schema extensions with the default datasource, then you will need to define a virtual datasource whose parentDatasource is the default datasource.

Even if you choose not to use schemas with the default datasource, you may still want to define a virtual datasource so that your application has a consistent, unique name to use to access the managed datasource. If you later choose to use a different database than the default, you can still use the same data source name to access the database, but have the virtual datasource refer to a different parent data source.

The default managed datasource provides for an encrypted database stored within the workspace directory. Multiple applications could share the single instance of the database, by defining schemas within the database to provide unique to the table references. If the configuration of the default datasource is not suitable, then you will need to define a new datasource using the extension. Virtual datasources do not allow you to change the database configuration, such as encryption, or location.

One of the main values of using the Managed Datasource is the definitions of the schemas that can be associated with the virtual or actual datasource definitions. If you choose not to use the schema capability, you can still use the Managed Datasource and create your own code to perform definition, migration, and population tasks.

Iterative development

If you are using managed data sources with schemas, to iteratively update the database, it is recommended that you provide updates in the form of incremented schema versions. The database management life cycle framework will properly update and migrate your schema once the managed data source is accessed. As an administrator, it is expected that you work with the developer that is using the managed data source to make sure that all the data required by the developer is in the managed data source. An example workflow would be this:

1. Administrator deploys a managed data source jdbc/MyDataSource with schema version 1.0.
2. Developer uses the jdbc/MyDataSource but finds later on that the managed data source needs another table and contacts the administrator.
3. The administrator updates the schema to version 2.0 and adds a task to create a new table as the developer requested.
4. Developer uses the jdbc/MyDataSource again and finds that the managed data source contains the table.

Proper interactions between an administrator and developer insures that the managed data sources are at a state that developers want them in.

In terms of testing, it is recommended that administrators deploy their updates to a new managed data source for testing before deploying to the original data source. This will ensure that when the old managed data source is updated, developers won’t see any problems. It is important to interact with a developer during this testing period to ensure the developer understands what managed data sources should be used in testing and production.
Security considerations for managed data sources

Depending upon the managed data source definition, the data source may use the platform keystore to store password or encryption key information. By associating a particular platform keystore alias with a data source connection attribute, such as a bootPassword for the Derby managed data source provider, the information can be securely stored within a keystore. Only after the appropriate password is used to unlock the platform keystore, can the database password or encryption key information be obtained.

If the keystore alias is allowed to be created on first access with a random value, it is important to retain both the keystore and the password to the keystore to continue access to the database. If the keystore is no longer accessible, then the database may be completely inaccessible, since the necessary keys are available only in the keystore.

The value associated with the alias needs to remain constant from the initial creation, as the managed data source component does not currently provide a mechanism to change the password or encryption key associated with a database.

If the database needs to be accessed external to the managed data source component, or outside of the Lotus Expeditor runtime, and aliases to the platform keystore are used to contain the password, then applications must first access the keystore to obtain the appropriate values to supply when accessing the database.

Refer to "Managing Secure KeyStore" on page 100 for more information on accessing the Secure KeyStore.

Testing notes

When testing managed data sources and working with new managed data sources, you may find yourself in situations where there were errors due to a bad script or database task. Sometimes it is necessary to clear out managed data sources you have been working with and to start from a clean slate. IBM Lotus Expeditor stores all databases in the workspace directory (mostly under the db directory unless specified otherwise). To start fresh, it is necessary to delete the directory in which the databases you are interested in reside. This will effectively clear out the cache the managed data source uses for its databases and schemas. Once this is completed, you are effectively starting with a fresh system that will act like no managed data sources have been accessed yet. This series of steps can be very useful when testing new managed data sources before deploying them. If troubles arise from testing managed data source, it is recommended to look at the logs for important error information.

Related information

"Defining a managed data source” on page 132
You can define a managed data source, using the Eclipse extension points and the IBM Lotus Expeditor Toolkit.

"Defining an encrypted managed data source” on page 132
You can define an encrypted managed data source, using the Eclipse extension points and the IBM Lotus Expeditor Toolkit.

"Defining an encrypted managed data source with authentication” on page 132
You can define an encrypted managed data source that uses the platform credential store for authentication.

"Defining a database” on page 133
The value in using the Managed Datasource capability is that the defined scripts and tasks are run automatically during the initial datasouce access. This eliminates the requirement for application code to handle tasks such as database definition, population and migration tasks.

"Defining a database schema for initial population using a database task” on page 133
You can define a schema for data population using a script.
Defining a managed data source
You can define a managed data source, using the Eclipse extension points and the IBM Lotus Expeditor Toolkit.

Follow these steps to define a managed data source:
1. Using the IBM Lotus Expeditor Toolkit, define a managed data source by specifying the 
   com.ibm.rcp.database.core.datasource extension point.
   • To define a normal managed data source, use the datasource element.
   • To define a managed virtual data source, use the virtualDatasource element. Virtual data sources
     are powerful as they let you reuse other data sources (similar to the default data source) but give
     you the ability to provide your own schemas and tasks.
2. Define a valid JNDI binding to match the data source's JNDI name using the
   com.ibm.pvc.jndi.provider.java.binding extension point.

After these steps are performed, the managed data source is defined.

Defining an encrypted managed data source
You can define an encrypted managed data source, using the Eclipse extension points and the IBM Lotus Expeditor Toolkit.

Follow these steps to define a managed data source:
1. Using the IBM Lotus Expeditor Toolkit, define a manage data source by specifying the
   com.ibm.rcp.database.core.datasource extension point.
2. Use the existing Apache Derby database provider and specify the following connection encryption
   attributes:

   **Table 16. Apache Derby Encryption attributes**

<table>
<thead>
<tr>
<th>Encryption attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
   | encryptionProvider   | Specifies the provider for data encryption. If none is specified, the JRE's
   |                      | provider will be used. |
   | encryptionAlgorithm  | Specifies the algorithm for data encryption. If none is specified, the default
   |                      | value is DES/CBC/NoPadding. |
   | dataEncryption       | Specifies data encryption on disk for a new database. |

   Refer to the Apache Derby documentation for more information about specifying encryption attributes.
3. Define a valid JNDI binding to match the data source’s JNDI name using the
   com.ibm.pvc.jndi.provider.java.binding extension point.

The encrypted managed data source is defined.

Defining an encrypted managed data source with authentication
You can define an encrypted managed data source that uses the platform credential store for authentication.

Follow these steps to define a managed data source:
1. Using the IBM Lotus Expeditor Toolkit, define a manage data source by specifying the
   com.ibm.rcp.database.core.datasource extension point.
2. Use the existing Apache Derby database provider and provide the following connection encryption and credential alias attributes:

<table>
<thead>
<tr>
<th>Encryption attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>encryptionProvider</td>
<td>Specifies the provider for data encryption. If none is specified, the JRE’s provider will be used.</td>
</tr>
<tr>
<td>encryptionAlgorithm</td>
<td>Specifies the algorithm for data encryption. If none is specified, the default value is DES/CBC/NoPadding.</td>
</tr>
<tr>
<td>dataEncryption</td>
<td>Specifies data encryption on disk for a new database.</td>
</tr>
<tr>
<td>credential</td>
<td>Specifies the credential alias used by the managed data source to boot the database.</td>
</tr>
</tbody>
</table>

Refer to the Apache Derby documentation for more information about specifying encryption attributes.

3. Define a valid JNDI binding to match the data source’s JNDI name using the com.ibm.pvc.jndi.provider.java.binding extension point.

The encrypted managed data source is defined.

**Defining a database**

The value in using the Managed Datasource capability is that the defined scripts and tasks are run automatically during the initial datasource access. This eliminates the requirement for application code to handle tasks such as database definition, population and migration tasks.

To use this capability of the Managed Datasource, you must define a schema and associate it with a managed datasource.

1. Create the SQL script files, or create the classes (that implement the com.ibm.rcp.database.core.DatabaseTask) that will operate on the database.
2. Define the com.ibm.rcp.database.core.schema extension with a valid schema name and version.
3. Add create, migrate, and populate definitions to the extension, referring to the scripts or classes created in step 1).
4. Include a schemaReference in the target managed datasource (com.ibm.rcp.database.core.datasource) extension. If you are defining a new managed datasource, you can directly include the schema reference. If you are using the default managed datasource, you will need to first define a virtual datasource on the default datasource, and include the schema reference in the virtual datasource definition.

When the application code uses JNDI to look up the managed datasource, the steps associated with the schema will performed prior to the datasource being returned to the calling code.

**Defining a database schema for initial population using a database task**

You can define a schema for data population using a script.

To define a schema for data population using a database task:

1. Implement the com.ibm.rcp.database.core.schema extension point using the IBM Lotus Expeditor Toolkit with a valid schema version and database task that uses the populate action.
2. Locate the data source you want to populate and add the proper schemaReferences.

The managed data source is then populated.

**Defining a database provider**

You can define a database provider.
To define a database provider, implement the com.ibm.rcp.database.core.provider extension point with a valid database provider ID and a class that implements the DatabaseProvider interface.

**Migrating a managed data source to a new schema**
You can migrate a managed data source to a new schema.

To migrate a managed data source to a new schema:
1. Implement the com.ibm.rcp.database.core.schema extension point with the same schema name as the previously installed schema. The only difference is that the version number must be greater than the previously installed schema. This allows the database catalog to recognize that an updated schema exists and it can be applied to the data source.
2. Implement the DatabaseTask interface to provide code that migrates the database.

**Using the Network layer**
The network layer is a base network management layer in Lotus Expeditor Client that provides a framework to applications and platform components for network invocations and network error handling.

It provides users with the capability to determine the current status of the client platform as well as their connectivity to remote servers, and HTTP resources including Web services. The application can choose to be notified of client status changes or to check the client platform status by using the public APIs for offline manager. The application can handle the network error by using the public APIs for Net Faults. The application can also check the real network status by using the public APIs for network status. This layer also includes the Rich Client Platform (RCP) version of HTTP URL Handler which integrates with account API and offline manager.

**Adding and configuring customized handlers**
As administrator, you can add handlers and configure the platform to use its customized handlers.

You must create the handler and add the handler to the platform. See "Adding and configuring customized handlers" in Developing Applications for Lotus Expeditor for more information about creating a handler and adding the extension to a plugin.xml.

**Extending the platform for Device Manager properties management**
The Device Manager server working with the Enterprise Management Agent on the client can update individual properties present in Java properties style files.

This allows for individual property updates to be distributed to multiple clients, without having to do a wholesale file replacement, which could result in incorrect settings being propagated to clients.

Files to be managed must be able to be processed as if they are ordinary Java properties files and be read and handled by the Java java.util.Properties class. Keys that contain characters that are not typically replaced in HTTP URI handling, such as the back slash (\) or plus sign (+), will be escaped using typical escape notation of using %xx where xx is the hex value of the character. When they are displayed on the Device Manager server, the escaped formatting characters will be displayed.

The platform enables four files for management by default. These files are the rcpinstall.properties and config.ini in the configuration directory and the rcplauncher.properties and plugin_customization.ini in the <installdir>/rcp directory.

To enable additional properties files to be updated, you must first register them with the platform. To register a new properties file, add a new extension to the com.ibm.rcp.props.dm.managedFile extension point. To do so, follow these steps:
1. If you have an existing project to which you want to add this extension point, go to step 3.
2. Create a new project to contain the extension:
   a. Create a new client services (or plug-in) project.
   b. If you intend to add code to this project, leave the default settings. If you do not intend to
      add code to this project, clear the Create Java Project option on the dialog page.
3. Go to the Extensions page in the Plug-in Manifest Editor for this project.
4. Click Add.
5. Clear the option Show only extension points from the required plug-ins.
6. Enter com.ibm.rcp.props.dm.managedFile as the Extension Point filter. Select this extension point
   and click Finish.
7. Optionally, add com.ibm.rcp.props.dm as a required bundle for project.
8. Optionally, fill in the ID and Name fields in the Extension Point details.
9. Select the extension that you have just added, right-click, then select New + managedFile.
10. In the extension details, complete the following details:
    a. In the file field, enter the location of the file. You can use variables in the location that will be
       replaced during runtime execution. Refer to the extension point documentation for more
       information. From the extension point, you can right-click, then select Show Description to
       display the description.
    b. In the name field, enter a name to identify this file. This can be the same as the file name, or you
       can use an alias to represent the file. This name is used during operations performed at the
       Device Manager server.
11. Save the file contents.

You can continue to add code to this plug-in just as with any other plug-in. Deployment of this plug-in
    follows the standard patterns. See Packaging applications for deployment on page 144 for more
    information.

Once the feature containing this plug-in has been deployed to the client, you can then initiate job requests
from the Device Manager server. See Managing properties files on page 203 for more information.

**Adding a managed settings provider**

A managed settings provider is an Eclipse plug-in that knows how to pull settings from an associated
back-end management system and pass them to the managed settings framework in the appropriate
format.

IBM Lotus Expeditor provides providers for settings specified by the Portal Policy Manager, for settings
that are specified in XML files and for policies specified in Domino (shipped with Notes 8.0). If you
would like to control preference or policy settings from a different back-end system you can have a
developer create a custom provider for you.

Custom providers must do the following:
1. Extend the ManagedSettingsProvider extension point provided by the plugin
   com.ibm.rcp.managedsettings. Additional information is provided in the schema of
   the com.ibm.rcp.managedsettings plug-in.
   For example:
   
   ```xml
   <extension point="com.ibm.rcp.managedsettings.ManagedSettingsProvider">
     <provider>
       class="com.isv.Provider"
       weight="10"/>
   </extension>
   ```

   The class attribute points to a class that implements the interface
The weight attribute is intended to resolve conflicts between multiple providers providing the same data. This value may be overridden by the administrator. The values provided by the provider with the lower weight get priority. Two providers could unintentionally be managing the same setting group if, for example, you initially set up mail policies in an XML file but decided to move them to Portal and forgot to delete them from the XML file. The weights for known providers have the following default values, which the administrator can change:

```java
com.ibm.rcp.managedsettings.provider.file = 300
com.ibm.rcp.managedsettings.provider.portal = 200
com.ibm.notes.managedsettings.provider = 100
```

2. Implement the `com.ibm.rcp.managedsettings.provider.Provider` interface. Javadoc can be found in `Developing Applications for Lotus Expeditor → Reference → Javadoc → Managed Settings`.

At the beginning of a managed settings update, the framework calls `getProvidedSettingGroups()` to determine which setting groups the providers currently provides data for. This is used to identify which setting groups have been deleted since the last update.

To implement this method, you will need to determine how the concept of a setting group relates to the back-end system that you are writing the provider for. A setting group is a group of related settings. It is usually the level of granularity at which settings modifications are tracked, in the event that modifications are not tracked at the individual setting level. If the back-end system does not have the concept of a setting group, you can just hard-code a single value and put all your settings in this setting group.

For example: Retrieves the names of the Setting Groups that this provider knows about.

**Returns:**

Collection of Strings containing the names of the Setting Groups that this provider can provide.

**Throws:**

ProviderException if the call failed for any reason.

```java
public Collection getProvidedSettingGroups() throws ProviderException {
    ArrayList settingGroups = new ArrayList();
    String[] providedSettingGroups = BackEndSystem.getPolicyNames();
    ArrayList ret = new ArrayList();
    for (int i = 0; i < providedSettingGroups.length; i++) {
        ret.add(providedSettingGroups[i]);
    }
    return settingGroups;
}
```

Setting groups are represented by the `SettingGroup` class, which is a data structure that enables applications to easily pass settings data to the Managed Settings Framework. The `SettingGroup` class is used to associate a set of name-value pairs, indicate their locked or unlocked state, and note the time that any of the values were last changed by the administrator. After calling `getProvidedSettingGroups()`, the framework calls `getUpdates()` and passes it a `Map` which specifies the setting groups and the last known modified dates of those setting groups. The method should return a list of `SettingGroup` objects for any of the setting groups that have changed since the last modification date. The Managed Settings Framework then stores the returned `SettingGroup` objects in the local store on the client. For example, if one of the entries in the map is setting group A, with a last modification date of January 1, 2006, the provider should return a `SettingGroup` object in the list if any of the values of the settings in setting group A have changed since January 1, 2006.
To be stored in the correct place in the Eclipse Preference store, the settings and setting groups must use the naming conventions described in “Specifying settings” on page 71. If you are unable to specify the settings that way due to limitations in the back-end system, this must be encoded in the provider instead.

If any updated setting group is missing any settings that were provided in the previous update, those settings are treated as newly unmanaged and will be unlocked.

Example:

```java
public List getUpdates(Map updateMap) throws ProviderException {
    ArrayList ret = new ArrayList();
    Iterator groups = updateMap.keySet().iterator();
    while (groups.hasNext()) {
        String policySet = (String) groups.next();
        Date requestChangesSince = updateMap.get(policySet);
        Map settingsChanges = BackEndSystem.getPolicyChanges(policySet, requestChangesSince);
        SettingGroup settingGroup = new SettingGroup(policySet, BackEndSystem.getLastChangeDate(policySet));
        Iterator settingsList = settingsChanges.entrySet().iterator();
        while (settingsList.hasNext()) {
            Map.Entry setting = (Map.Entry) settingsList.next();
            settingGroup.add(setting.getKey(), setting.getValue(), true);
            ret.add(settingGroup);
        }
    }
    return ret;
}
```

You may wish to combine your implementations of getProvidedSettingGroups() and getUpdates() to require just one call to the server by fetching the information required by both in the getProvidedSettingGroups() call and then caching the update information so that it can be later returned by the getUpdates() method. The only issue with doing this is that at the time getProvidedSettingGroups() is called, you do not have access to the last modification dates provided as an argument to the getUpdates() method. You can get access to the last modification dates any time you want by calling ProviderHelper.getSettingGroupLastModDates(String providerId).

You can use the Accounts API to narrow down the list of settings and Setting Groups to ones that applies to the logged in user.

Optional: If the provider knows when it has updates, it can request that the Managed Settings Framework run an update immediately instead of waiting for the update interval to expire. The Managed Settings Framework calls Provider.addProviderUpdateListener on the provider to register for these notifications. Implement this method in order to track the listeners. When the provider knows it has updates, it should call the onRequestUpdate() method on the registered listeners and an update will be run immediately.

**Defining accounts for Apache Axis WebServices clients**

You can programmatically create accounts for Apache Axis WebServices clients.

Refer to “Programmatically creating accounts for Apache Axis Web Services clients” in Developing Applications for Lotus Expeditor.

**Defining locations**

You can define new locations, using the Eclipse extension points and the IBM Lotus Expeditor Toolkit. If there are no location contributions, IBM Lotus Expeditor will provide two default locations called Online and Offline.

To define a new location, implement the com.ibm.rcp.locationmanager.locationActionSets extension. Make sure that you provide the preferred Online or Offline state associated with your new location by
specifying the offline attribute. The user will be able to switch between locations via UI controls. In addition, you can use com.ibm.rcp.locationmanager.LocationManager public API to switch between locations programmatically.

The following is an example of a location action set. In this example, a location called Office is contributed to the platform. The preferred platform state for this location is online (note the sub-elements and the way attributes are used):

```xml
<extension
  point="com.ibm.rcp.locationmanager.locationActionSets">
  <actionSet
    automaticStartup="false"
    id="com.ibm.rcp.locationmanager.test.locations"
    label="com.ibm.rcp.locationmanager.test.locations"
  >
    <action
      class="com.ibm.rcp.locationmanager.test.actions.TestAction"
      id="com.ibm.rcp.locationmanager.test.office"
      label="Office"
      offline="false"
      tooltip="Switch location"/>
  </actionSet>
</extension>
```

**Defining component wiring**

Refer to the Portal information center for information about component wiring.

The Portal information center is located at [http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp](http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp)

**Extending the Text Analyzer**

You can extend the Text Analyzer framework to provide Spell Checker, which is used to check misspelled words in a document.

Spell Checker supports 26 languages and can be used by many editors, by implementing the document interfaces. You can contribute new engines and dictionaries to the Text Analyzer framework so that applications can use customized engines and dictionaries through the framework.

**Table 18. Supported dictionaries**

<table>
<thead>
<tr>
<th>Languages</th>
<th>Supported dictionaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>English (Australian)</td>
</tr>
<tr>
<td></td>
<td>English (Great Britain)</td>
</tr>
<tr>
<td></td>
<td>English(United States)</td>
</tr>
<tr>
<td>Chinese (simplified)</td>
<td>Chinese (Simplified)</td>
</tr>
<tr>
<td>Chinese (traditional)</td>
<td>Chinese (Traditional)</td>
</tr>
<tr>
<td>French</td>
<td>French (French)</td>
</tr>
<tr>
<td></td>
<td>French (Canadian)</td>
</tr>
<tr>
<td>German</td>
<td>German (Swiss)</td>
</tr>
<tr>
<td></td>
<td>German (Germany reform)</td>
</tr>
<tr>
<td>Italian</td>
<td>Italian</td>
</tr>
<tr>
<td>Spanish</td>
<td>Spain</td>
</tr>
<tr>
<td>Portuguese (Brazilian)</td>
<td>Portuguese (Brazilian)</td>
</tr>
<tr>
<td>Arabic</td>
<td>Arabic (Arabic)</td>
</tr>
</tbody>
</table>
Table 18. Supported dictionaries (continued)

<table>
<thead>
<tr>
<th>Languages</th>
<th>Supported dictionaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>Czech (Czech Republic)</td>
</tr>
<tr>
<td>Danish</td>
<td>Danish (Denmark)</td>
</tr>
<tr>
<td>Dutch</td>
<td>Dutch (Holland)</td>
</tr>
<tr>
<td>Finnish</td>
<td>Finnish (Finland)</td>
</tr>
<tr>
<td>Greek</td>
<td>Greek (Greece)</td>
</tr>
<tr>
<td>Hebrew</td>
<td>Hebrew (Israel)</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Hungarian</td>
</tr>
<tr>
<td>Norwegian</td>
<td>Norwegian (Bokmal)</td>
</tr>
<tr>
<td></td>
<td>Norwegian (Nynorsk)</td>
</tr>
<tr>
<td>Polish</td>
<td>Polish</td>
</tr>
<tr>
<td>Portuguese</td>
<td>Portuguese (Portuguese)</td>
</tr>
<tr>
<td>Russian</td>
<td>Russian (Russia)</td>
</tr>
<tr>
<td>Swedish</td>
<td>Swedish (Sweden)</td>
</tr>
<tr>
<td>Turkish</td>
<td>Turkish (Turkey)</td>
</tr>
<tr>
<td>Catalan</td>
<td>Catalan (Catalan)</td>
</tr>
</tbody>
</table>

**Contributing a custom dictionary**

Use the com.ibm.rcp.textanalyzer.Dictionaries extension point for applications to contribute new dictionaries.

A plug-in that needs to contribute a dictionary into the framework could define the extension, similar to the following:

```xml
<plugin>
  <extension
    point="com.ibm.rcp.textanalyzer.Dictionaries">
    <dictionary
      description="English medical dictionary"
      dict_name="en_US_medical.dic"
      engine="jFrost"
      filePath="./dictionaries/en_US_medical.dic"
      language="en-US"
      locale="en-US"
      provider="IBM"
      version_info="1.0.0"
      type="spell"/>
  </extension>
</plugin>
```

In this example, the plug-in contributes a medical dictionary in English to the existing IBM jFrost engine, and the dictionary file position is in the dictionaries folder of that plug-in. For more information, refer to "Contributing custom dictionaries" in Developing Applications for Lotus Expeditor.

**Contributing a new engine**

Use the com.ibm.rcp.textanalyzer.Engines extension point for applications to contribute new engines.

A plug-in that needs to contribute a spell checking engine to the framework can define the extension and implement the interface com.ibm.rcp.textanalyzer.spellchecker.SpellCheckerEngine. For more information, refer to "User dictionary manager" in Developing Applications for Lotus Expeditor.
Deploying the IBM Lotus Expeditor micro broker

There is only a single micro broker instance per platform. The micro broker instance is created programmatically by using the administration APIs.

The micro broker instance once created will be started automatically by the com.ibm.micro bundle. Since the com.ibm.micro bundle will be lazy started as a result of calling the administrative APIs on the micro broker, the only bundle that needs to be added to the platform or application lifecycle is your application bundle that checks for existence of or creates a micro broker instance. See “Managing life cycle” on page 108 for more information.

To begin initial prototyping using the micro broker, a IBM Lotus Expeditor micro broker Getting Started feature is provided on the updates\platform update site. This feature is not installed by default and must be installed using one of the provisioning mechanisms. To use this feature, either add the com.ibm.micro.gettingstarted plug-in to the platform life cycle to perform the initial configuration of the micro broker instance, or use the platform console to manually start this plug-in. The micro broker instance is named FirstBroker. This plug-in is for evaluation and initial prototyping use and provides a very basic configuration. You may need to create your own broker instance specific to your application requirements, configuring capabilities such as connections and connection types, persistence, connection factories and more.

For more information, refer to “Configuring and administering micro broker components” in the micro broker documentation for more details.

Creating client runtime images

This section describes how to use the Platform Builder tool to create custom runtime images of the Lotus Expeditor Client platform.

Getting started building platforms

Use the Platform Builder tool to create a custom runtime image of the Lotus Expeditor Client platform.

To get started, use the Product Configuration Wizard (File > New > Other > Plug-in Development > Product Configuration) to guide you through the process of building an initial runtime image. Choose which plug-ins, bundles, features, and application components you want so that you can build the smallest runtime footprint. If you want to refine the runtime image at a later date, you can use the Product Configuration Editor to rebuild the image.

Follow these basic steps to build a runtime image:

<table>
<thead>
<tr>
<th>Step</th>
<th>Refer to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ensure that your development system meets all software and hardware requirements and that you are aware of any Platform Builder limitations at the time of this release.</td>
</tr>
<tr>
<td>2</td>
<td>Set the plug-in development target platform location to the Lotus Expeditor Client runtime.</td>
</tr>
<tr>
<td>3</td>
<td>Create/update the configuration file, if necessary. When the configuration is created, the file name is defined in the Product Configuration wizard with a .product extension by default.</td>
</tr>
<tr>
<td>4</td>
<td>Export the target runtime image to a local directory.</td>
</tr>
<tr>
<td>5</td>
<td>Run the target runtime image.</td>
</tr>
</tbody>
</table>
Attention: Use of runtime platforms generated with the IBM Lotus Expeditor Toolkit in a production environment requires a separate license. Contact your IBM sales representative for deployment licensing.

Setting up the target platform
To enable the plug-ins provided with the Lotus Expeditor Client, you must re-target your workspace to use Lotus Expeditor Client runtime platform.

Because of this requirement, it is recommended that you create a new workspace to use when you develop applications for this product.

Creating a Platform Builder configuration
A Platform Builder configuration can reside within any project in the workspace, however, it is suggested that a project, such as a General Project, be created as the location where the Platform Builder configuration is saved.

To create a General Project, follow these steps:
1. Select File → New → Project → General → Project and click Next.
2. Enter the project name, then select Finish.

To create a Platform Builder configuration, follow these steps:
1. Select File → New → Other → Plug-in Development → Product Configuration and click Next.
2. Enter or select the parent folder.
   The parent folder name is the name of the project where the Platform Builder configuration will reside.
3. Enter or select a file name for the Platform Builder configuration. The file name extension should be left as .product.
4. Select Finish.
   The content of the product file can be initialized using one of the following three methods. The option Create a configuration file with basic settings creates a basic product configuration file with minimal settings. All information specific to the product configuration must be added using the Product Configuration Editor. The option Use an existing product creates a product configuration using information such as Product ID and Application from the selected extension. Additional information can then be added using the Product Configuration Editor. The option Use a launch configuration initializes the contents of the product file using the information defined in an existing Client Services launch configuration. Information can be modified using the Product Configuration Editor.
5. Open the .product file using the Product Configuration Editor.
   When the file is created, it will be opened by default in the editor. Subsequently, the file can be opened using the standard Eclipse techniques of double clicking the file, or right-clicking the file name and selecting Open With > Product Configuration Editor from the context menu displayed.
6. The information contained in the Platform Builder configuration is displayed within the Product Configuration Editor on four pages. The four pages are labeled Overview, Configuration, Launcher and Branding. To select a page, click on the page name. The information on all four pages should be filled in before attempting to launch (test) or export the Platform Builder configuration.
7. On the Overview page:
   a. Enter the product name.
      This name will be used to identify this Platform Builder configuration. The name is also used as the name= parameter in the .eclipseproduct file that will be exported for use by Lotus Expeditor Client.
   b. Select the Target configuration.
      Each target configuration identifies the initial set of Lotus Expeditor Client features and plug-ins that should be used. Optional features and plug-ins can then be added, as well as workspace
features and plug-ins. Platform Builder supports launch and export of runtime images intended for the desktop. For this reason, the Default Device Target should not be used with Platform Builder.

c. Select the product ID and/or application.
   The Target configuration contains a recommendation for the product ID or application to be used. The values recommended can be modified if desired.

d. Select the Target Environment.
   This indicates whether a runtime image is desired for use on a Windows or Linux system.

e. Select the Virtual Machine (VM) that implements the Java specifications.
   Again, the Target configuration contains a recommendation for the VM to be used with the configuration. The value recommended can be modified if desired. The blank line should be selected from the list if a VM should not be included in the configuration. This is intended for use in cases where the system on which the runtime image installs already has an installed VM.

f. Select the appropriate radio button to choose if this configuration is based on plug-ins or features. The default is features.

8. On the Configuration page:

   a. Select Add to open the Feature List Selection dialog and add additional features when the configuration is based on features, or Plugin List Selection dialog for plug-ins when it is based on plug-ins. The list selection dialogs have two pages with the first page listing plug-ins (or features if feature-based) that are required or optional as defined by the target configuration chosen in the Overview page.

   6.1.2 The first page also provides a Select Recommended button, which allows you to include recommended, optional plug-ins (or features if feature-based) for the current target selected.

   The second page of the list selection dialog contains the rest of the plug-ins or features that were found by Eclipse, but are not on the first page of this dialog. Any plug-ins or features that are required for the selected target configuration appear selected and grayed out.

   Note: For a custom-created plug-in, you must manually add the required plug-ins used by extension points. This ensures that the Platform Builder includes all the required plug-ins when the product is exported.

   b. Select the languages to include in the platform.

   c. A config.ini file is created by default, which is used by the runtime during startup. Users can replace this file with their own custom config.ini file in the workspace by selecting Use an existing config.ini file and selecting it using the Browse button.

9. On the Launching page:

   a. Select the launcher name and launcher configuration file name if not using the standard Client Services launcher.

   b. Enter the program arguments, VM arguments, and bootstrap entries on the platform tab that corresponds to the Target Environment. Selecting the platform tab that corresponds to the Target Environment will provide a set of recommended values.

10. On the Branding page, select the appropriate personality.

11. Save the configuration.

To execute the Lotus Expeditor Client Platform in a test environment, switch to the Overview tab and select Launch the Product.

Note: You must test the runtime image on the platform it is intended for use on. For example, runtime images with a target environment of Windows, cannot be tested on a Linux system.
Exporting a target runtime image
Follow these steps to export a target runtime image.

To export a runtime image, perform the following procedure:
1. Select the Eclipse Product export wizard link, from the Overview page of the Product Configuration Editor, or select File > Export > Plug-in Development > Eclipse Product.
2. Enter the name of a directory or an Archive file name (only .zip extensions are supported).

After you export the runtime image (provided you created an Archive file), it can be unzipped to a directory on the system where it will run using local tools such as pkzip or gzip.

Note: If you encounter any synchronization errors about plug-ins not being in the workspace during the export process, clear the synchronize before exporting field and try exporting again.

Migrating a WebSphere Everyplace Deployment 6.0 Platform Builder project to Lotus Expeditor Client 6.1.x
Learn how migration automatically occurs when migrating a WebSphere Everyplace Deployment 6.0 Platform Builder project to Lotus Expeditor Client 6.1.1 or 6.1.2.

Migration will automatically occur under the following circumstances:
1. A workspace from the 6.0 release is opened using the tools from the 6.1.1 or 6.1.2 release
2. A project created using the 6.0 release is imported (using the standard Eclipse import wizard) into the workspace using the tools from the 6.1.1 or 6.1.2 release.

Running a target runtime image
Learn how to run a target runtime image.

To launch the target runtime image, do one of the following:
• On Windows systems, run the startup.bat file.
• On Linux systems, run the startup.sh file.

These files are created in the root directory of the exported image.

Note: Do not install a runtime image into a directory that contains a semicolon in the directory name.

The startup.bat file that Platform Builder exports contains two programs (WCTMELaunchUtil and rcplauncher.exe) that are executed to start their instance of the desktop client. WCTMELaunchUtil is used to initialize the environment. Among other things, this program creates a file required by the second program (rcplauncher.exe). The rcplauncher.exe program is used to start the client. Each time the first program is executed, a new unique workspace name is created. If you do not want a new workspace name created every time the startup.bat runs, edit the batch file to create two separate batch files.
• Save the first line of the startup.bat file to a separate batch file. Run this batch file only once to initialize the workspace.
• Save the second line of the startup.bat file to a separate batch file. Run this batch file to every time you want to start the client.

Configuring application layout
You define the layout of applications, such as views and editors, within the workbench using two methods.
The client platform provides technology known as Composite Application Infrastructure (CAI). CAI relies on a definition of the application layout contained in an XML stream. The XML stream may be obtained from a static definition deployed to the local file system, a definition contained within a plug-in, or may be retrieved from a Portal Server.

With an IBM extension, WebSphere Portal 5.1 supports cooperative portlets for JSR 168 portlets, in which one JSR 168 portlet can communicate with another JSR 168 portlet. With the WebSphere Portal V5.1.0.1 release, you can use cross-page wiring between cooperative portlets for both JSR 168 portlets and IBM API portlets.

See "Understanding composite applications" in Developing Applications for Lotus Expeditor for information about how to construct a CAI XML definition. See "Managing Rich Clients using the Portal server” on page 205 for information about how to layout applications on the Portal Server to dynamically generate XML to be processed on the client.

The other model for application layout is to use a programmatic/extension point based definition, which is part of the standard Eclipse programming model. This model requires the definition of perspectives to contain views and editors, and to manage the layout of the application. Both models can be mixed on the same client.

---

**Packaging applications for deployment**

This section describes how to deploy applications.

The term application is used in this section to refer to a group of one or more features that must be deployed together to provide some useful capabilities.

The deployment artifacts are plug-ins, fragments, features, and update sites.

Plug-ins contain the code and resources that provide specific functions. Fragments can contain code to extend other plug-ins, but are typically used to provide locale specific information or operating system specific implementations. Plug-ins and fragments are not individually installed into the client runtime, but must be packaged within a feature. A feature is the minimal install artifact. Features include plug-ins and fragments. Features may also include other features. Features can also require other plug-ins and features to be installed.

An update site is a repository of features. Features on an update site may or may not have any relationship one to another. Depending on the methods used for deployment, update sites are optional. To deploy features without adding them to an update site, you can use the following deployment choices:

- Feature distribution using the Device Manager Server
- Feature distribution using the Portal managed Client

To deploy features that are contained within an update site, you can use the following deployment choices:

- Local update site
- Remote update site
- Local archived update site
- Update site distribution using the Device Manager Server

To create features and update sites, use the following tasks:

- Creating a feature
- Creating an update site
- Exporting a feature and its plug-ins
Once the feature or update site is created, see [“Managing the client runtime” on page 181](#) for information on deploying the feature or update site.

**Understanding feature life cycle**

This section describes the life cycle of features for the IBM Lotus Expeditor platform. It is important to understand the feature life cycle when distributing features to client systems.

Just like the plug-ins that exist within the platform, the features that define the plug-ins have a similar life cycle that affects the ability to use what they provide.

A feature exists in one of three states:

**Uninstalled**

This refers to a feature that is not installed on the platform sites. This is either a feature that has never been installed or one that was previously installed and then had an uninstall operation performed on it. During the uninstall operation, the uninstall methods of the InstallHandler are invoked.

**Installed**

The installed state is entered either by performing an install action on an uninstalled feature, or disabling an enabled feature.

For an uninstalled feature that is installed, the install related methods of any InstallHandler associated with the feature are called.

For a previously enabled feature that is being disabled, the unconfigure related methods of any InstallHandler are called. For a feature to successfully install, the feature descriptor as well as any required plug-ins must be accessible and valid to be copied to the platform site. In addition, any requirements expressed by the feature must also be met before it can be installed. A validation phase occurs prior to attempting an install to verify that the feature deployment descriptors are accessible, and that the requirements are satisfied. Plug-in existence and validity (that is, the valid signed certificate) are checked during the actual task of installing the feature.

A feature that is only installed does not provide any function to the platform, and its contained plug-ins are not installed into the runtime.

**Configured (or Enabled)**

The configured (or enabled) state occurs by performing a configure (or enable) operation on a feature in the installed state. During the configure operation, the configure related methods of the InstallHandler are called to perform tasks.

Once configured, the feature and its contained plug-ins are then able to provide function to the platform.

A feature that is currently enabled can be disabled, but only if by doing so it would not invalidate other features dependencies upon the feature or its contained plug-ins.

The enable and disable operations require a platform restart before the results are effective.

Now that the life cycle of features has been explained, we can now discuss the role of the InstallHandler, which was referenced above.

InstallHandlers may be specific to either a single feature, or global to the entire platform. Regardless of the type, the behavior and call sequence is equivalent.

The InstallHandler class will be instantiated prior to any operation. The first call (after the constructor) will always be the initialize method. This method sets up the install handler and provides information that can be used throughout the call sequence.

During an install operation, the following sequence of calls would be made:
1. initialize()
2. installInitiated()
3. pluginsDownloaded()
4. nonPluginDataDownloaded()
5. completeInstall()
6. installCompleted()

Refer to the javadoc for specific signature information. Once initialize method has been called, the installCompleted call will always be performed. The installCompleted call is passed a Boolean indicating success or failure of the install operation.

Tasks that need to be performed to install the feature should be performed during the range of calls from installInitiated to completeInstall. A CoreException thrown during these methods will result in the feature failing to install. At the point when the installCompleted() method is called, the feature is either installed or not installed. While the signature allows a CoreException to be thrown at this point, it is strongly recommended that features do not throw a CoreException from the installCompleted (true) call, as it will result in error messages being logged, but will still result in the feature being installed.

During the configure (or enable) operation, the following sequence of calls would be made:
1. initialize()
2. configureInitiated()
3. completeConfigure()
4. configureCompleted()

As in the install case, once the install handler has been initialized, the configureCompleted() method will always be called. As in the install case, the configureCompleted() does allow a CoreException to be thrown, but work should preferably be done in either the configureInitiated or completeConfigure methods. A failure in any of the methods will result in the feature not being configured.

The install handler is also called during the disable operation in the following sequence:
1. initialize()
2. unconfigureInitiated()
3. completeUnconfigure()
4. unconfigureCompleted()

A CoreException thrown in the unconfigureInitiated method will abandon install handler processing for the feature, and the completeUnconfigure and unconfigureCompleted methods will not be called. Unless exceptions otherwise inhibit the call, unconfigureCompleted() will be called following an initialize. It is suggested that the main install handler work be performed in the unconfigureInitiated() and completeUnconfigure() methods.

Finally, the install handler is called during the uninstallation of a feature.
1. initialize()
2. uninstallInitiated()
3. completeUninstall()
4. uninstallCompleted()

A CoreException thrown in the uninstallInitiated method will abandon install handler processing for the feature, and the completeUninstall and uninstallCompleted methods will not be called. Unless exceptions otherwise inhibit the call, uninstallCompleted() will be called following an initialize. It is suggested that the main install handler work be performed in the uninstallInitiated() and completeUninstall() methods.
When the multiuser platform configuration is used, features installed by the administrator and available to the users will only be installed a single time. This means that the install methods of the install handlers will be run at most once. For this reason, and in anticipation of a change in a direction of install processing, it is not recommended that install handlers perform work related to feature configuration during the install related methods of the install handler. Features will be configured (or enabled) multiple times, at least once in each workspace, so the preferable location for feature related changes would be the configure related methods.

Creating a feature
To create a feature, use the Plug-in Development Environment within Eclipse or the Rational Software Development platform.

Prior to creating a feature, you should have the plug-ins and fragments that will be contained within the feature:
1. From your workspace, select File → New → Project → Plug-in Development → Feature Project
2. Enter the project name. This is typically the same name as the feature ID.
3. Enter the Feature ID, Feature Name, and Feature Version. The Feature Provider and Install Handler Library are optional.
4. Select Next.
5. Select the plug-ins and fragments that you want to add to the feature. If you do not want to enter them at this time, you can select Finish now, and add plug-ins and fragments later using the editor.
6. The editor opens. Use the editor tabs to specify additional information:
   - **Overview**
     - Enter feature descriptive information and supported environments.
   - **Information**
     - Enter description, license, copyright, and other update sites to visit.
   - **Plug-ins**
     - Enter the plug-ins and fragments that should be packaged, as well as supported environment information if needed.
   - **Include Features**
     - Enter the additional features to include, as well as supported environment information if needed.
   - **Dependencies**
     - Enter any dependency information that must be met before the feature will install.
   - **Installation**
     - Enter information such as install handler and other data files.
   - **Build**
     - Select the files that will be packaged into the feature.
7. Save the contents of the editor.


Specifying optional child features
The feature manifest (feature.xml) enables the definition of included features. Included features define a parent-child relationship for features, such that actions on the parent apply also to the child. For example, if the parent is installed, the child is also installed. An included feature may be marked as optional. This would enable a parent to be installed without some of the optional child features.

When the Lotus Expeditor platform installs a new feature, all optional children of the feature, if they are available on the update sites specified during the install operation, will be installed.
Controlling the feature install location
This section explains the available methods for controlling and specifying the target install site for a feature.

The platform provisioning system defines by default the following target sites into which features are installed:

eclipse
Contains the features and plug-ins that originate from the Eclipse platform.

rcp
Contains the features and plug-ins that are part of the client platform.

shared
Contains additional features and plug-ins intended for use by all users of the client platform.

“user” Located in the workspace, this target site contains additional features and plug-ins intended for use only by a single user of the client platform.

The target install site for a particular feature is determined by evaluating several conditions, including whether the feature is currently installed, colocation-affinity, ability to write a specific directory, and requested target site.

Colocation-affinity is an attribute that can be specified by each feature in the feature.xml descriptor. The colocation-affinity attribute specifies the ID of another feature. It directs the provisioning system to attempt to install the new feature into the same target site that the other feature (the feature whose ID is specified in the feature.xml). Using the Feature Manifest Editor from the Plug-in Development Environment, the colocation-affinity can be specified on the Installation page of the editor.

For example, suppose that feature with an ID of com.ibm.sample has been installed into the shared site. A new feature with an ID of com.ibm.extendedsample can then specify a colocation-affinity of "com.ibm.sample" to request that the feature is installed into the same location as com.ibm.sample:

```xml
<feature
   id="com.ibm.extendedsample"
   label="Extendedsample Feature"
   version="1.0.0"
   provider-name="IBM"
   colocation-affinity="com.ibm.sample">
</feature>
```

The platform provisioning system uses these decisions to determine the target site for a feature.

1. If a provisioning system detects a version of the feature that is already installed, the same install site will be used for any additional versions. This only applies when the feature has been installed into the Eclipse, RCP, or shared target sites.
   - If a deployer installs a feature X version 1.0.0 into the shared site, and later attempts to install feature X version 1.0.1, then version 1.0.1 will be installed into the shared site.
   - If a user A installs a feature Y into their user site in their workspace A, and later another user B attempts to install feature Y, feature Y will be installed a second time into the workspace B, since workspace A is not visible to user B.

2. If feature.xml specifies the colocation-affinity attribute, then the following decisions are made:
   a. If the feature that was specified in the colocation-affinity attribute is installed, then attempt to install into that site. If that site is not accessible to the user installing the feature, the feature installation will fail.
   b. If the feature that was specified in the colocation-affinity attribute is not installed, then continue with step 3 in the decision process.

3. If a specific target site is requested for the feature, then attempt to install the feature into this site. If that site is not accessible, then continue with step 5. A target site can be specified only for features distributed through the Eclipse feature distribution job using the Device Management server.
4. If the provisioning manifest or a ProvisioningRequest (provided through an API request) specifies 
   shared = true, then the shared site is the desired target site for installing the feature. If that site is not 
   accessible, then continue with step 5. (Refer to “Provisioning manifest DTD” on page 165 for more 
   information on the attributes available through the provisioning manifest.

5. If the provisioning manifest or a ProvisioningRequest object specifies shared = false, or if because of 
   accessibility issues, the desired target sites for step 3 and step 4 are not available, then attempt to 
   install the feature into the “user” site.

Note: The default setting of the shared attribute depends on the install configuration. If a multiuser 
   install configuration is being used, the default value of shared is true. If a single user install 
   configuration is being used, the default value of shared is false.

Since the colocation-affinity takes precedence over target site or shared attribute specification, specifying a 
colocation-affinity is the preferred model for controlling feature install locations. The client platform has 
defined the following features that can be used as targets for the colocation-affinity attribute:

com.ibm.rcp.site.anchor.shared.feature
   This feature is always installed into the shared site.

com.ibm.rcp.site.anchor.user.feature
   This feature is always installed into the user site.

If the platform is only ever used by a single user with a single workspace, then the target site for a 
feature is less important, since all features will always be available. In the multiuser configuration, the 
target install site becomes much more important, since some applications should be shared among all 
users of the platform, while other applications should be private to specific users.

As an example, assume a multiuser configuration in which deployers have admin rights to the shared 
directory, while ordinary users are limited to read-only rights. Feature W does not contain a 
colocation-affinity attribute. If a deployer attempts to install this feature, it will be installed into the 
shared directory, since this is the default option with no colocation-affinity and no target site specified. 
However, if a user attempted to install this feature, it would be installed into the user target site, and 
could not be shared by all users on the platform. The user site would be selected since the shared site is 
not accessible.

To limit the ability of a specific user to install feature W into their user site, the deployer could specify a 
colocation-affinity attribute with the ID “com.ibm.rcp.site.anchor.shared.feature”. The resultant site for 
a deployer would still be the shared site, chosen now not because it is the default, but because the 
colocation-affinity selects that site. A user attempting to install the feature would have the feature 
installation fail, since the site specified by the colocation affinity is not accessible.

Enabling features for update
This section provides information about how feature updates are handled. Updating an application is 
similar to the initial installation.

The Lotus Expeditor Client uses the eclipse and rcp/eclipse directories within the Lotus Expeditor 
Client 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.

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 features have a 
colocation-affinity attribute specified. In this case, the feature and its plug-ins are installed to the same 
location as the feature 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. A major version is a version that differs first at the major level. For example, using our version of 4.0.1, a version of 5.0.0 would be considered a major version upgrade.

The Automatic Updates capability provided as part of the Application Manager dialog enables updates of only major, 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>
```

For additional information on versioning, refer to the Feature manifest section of the Platform Plug-in Developer's Guide and to Eclipse [http://help.eclipse.org/help32/index.jsp].

**Eclipse feature update on devices**

You can search for updates to one or all existing applications that are installed on the Lotus Expeditor Client on the device. When new versions of features are available, they are installed into the same directory as the previous version. The installation directory for a feature update cannot be changed.

The default update type on the Lotus Expeditor Client for Devices is equivalent. This means that the update process will search for all equivalent versions on the update site and update to the highest of all versions. The update type can not be changed by the user. However, the administrator can change it using a properties configuration job. To change the update type, modify the following property:

```
./Configuration/Properties/config.ini/Eclipse.update.core.updateVersions
```

where config.ini is the file name, and Eclipse.update.core.updateVersions is the property key. There are three supported property values: equivalent, compatible, and major. The equivalent value means the update process only searches equivalent versions on the update site. The compatible value means the update process searches all equivalent and compatible versions, and the major value means the update process will search all versions greater than the existing one.

For more information about editing the properties, see Properties editing jobs in Using Lotus Expeditor Server.

**Exporting a feature and its contents**

Once the feature is defined, to deploy the feature, you will need to export the feature and its plug-ins.

If you will be creating an update site, skip to “Creating an update site” on page 151.

If you will be just deploying the feature, then you will need to build and export the feature and its contents. Select File ➤ Export ➤ Plug-in Development ➤ Deployable Features.

Select the features to export, enter a directory or archive file, then select Finish. The feature contents, as well as the included plug-ins, fragments, and features will be exported to the desired directory or archive file.

Creating an update site

An update site project can be created either inside your current workspace, or you can create it anyplace on the local file system.

Follow these steps to create an update site
1. Select File → New → Project → Plug-in Development → Update Site Project.
2. Enter the project name, then select Finish.
3. The Site Manifest Editor opens, enabling you to add features to the site. Select Add Feature to add features to the site.
   You only need to add the features that you specifically want to make available for installation. Features included in other features do not need to be added here unless you want to make them separately installable.
4. Once you have added all the features, select Build or Build All to build the features and the feature content.


Using the global install handler

Install handlers are Java classes that are run by the Update Manager during the install, configuration, and uninstall processing of a feature.

Update manager can handle standard installation of features, plug-ins, and fragments without writing any Java code. However, if additional tasks are required to be performed during install, configuration, and uninstall processing, an install handler can be used to perform these tasks. An example of tasks that might be performed by install handlers include configuration file updates.

The client platform provides an install handler that can be used to perform updates to files present on the client system. The install handler provides additional capabilities beyond that offered by the default install handler provided by Eclipse. Capabilities provided by the install handler include the following:

- Ability to modify rcpinstall:properties settings for single or multi-user for:
  - bootclasspath
  - PATH / LD_LIBRARY_PATH
  - class path
  - VM System properties
  - VM
  - Other properties
- Ability to set locations of Virtual Machine (VM) that implements the Java specifications, RCP base location, RCP launcher location, provisioning manifest location and version
- Ability to launch a native script or executable

Creating the install handler instructions file

The install handler processes instructions from a file named handler:properties that is present within the feature.

The handler:properties files should not contain absolute path references, as the actual file location may change depending upon either the launch directory or the workspace directory. The available substitution parameters should be used instead, as these will correctly handle the changes to ensure correct operation of the platform when either the launch location or workspace directory location changes.
All normal commands ("exec.command"-related commands) get executed during the completeConfigure() phase of installation. It is also important to note that most commands ("exec.command"-related commands) should properly remove their actions during the completeUnconfigure() phase of installation.

**Xbootclasspath.prepend=<path>**
**Xbootclasspath.append=<path>**

Adds all listed entries to the Xbootclasspath.prepend or Xbootclasspath.append properties in rcpinstall.properties.

Used by the native launcher to set the vm option -Xbootclasspath.

**library.path.prepend=<path>**
**library.path.append=<path>**

Adds all listed entries to the library.path.prepend or library.path.append properties in rcpinstall.properties

Modifies PATH on Windows or LD_LIBRARY_PATH on Linux.

**-D<pair>**

Adds value-pair settings to add VM system properties definitions to rcpinstall.properties.

Used by the native launcher to add settings to the vm.

**vm**

Adds vm setting to rcpinstall.properties.

Used by the native launcher to add entries to the system path variable.

**jvm.location=<value>**

Modifies the jvm location in rcplauncher.properties.

**rcp.base.location=<value>**

Modifies the rcp base location in rcplauncher.properties.

**rcp.launcher.location=<value>**

Modifies the rcp launcher location in rcplauncher.properties.

**exec.command.<arch>=<command>**
**u.exec.command.<arch>=<command>**

Executes a command using the Runtime.exec() from the feature directory. exec.command is processed during the feature enable processing. u.exec.command is processed during the feature disable processing. If an argument in the <command> contains spaces, then include the argument in quotation marks ")(". The exec.command/u.exec.command can be useful for running scripts or registering Windows services. For example:

- exec.command.win32x86=regsvr32 /s my.dll
- exec.command.linuxx86=sh -c myscript.sh
- exec.command.win32x86="C:\Program Files\SomeProgram\Program.exe" "C:\Documents and Settings\user\Application Data"

**mergemanifest=<value>**

Merges the file specified by <value> into the current provisioning manifest identified by rcpinstall.properties. <value> is expected to be a file system location for a new provisioning manifest file. Examples of the mergemanifest command are as follows:

- mergemanifest=$(plugin.dir)/com.ibm.rcp.newmanifest_1.0.0/upgrade.xml
- mergemanifest=$(features.dir)/com.ibm.rcp.newmanifest.feature_1.1.0/install.xml

See "Understanding the merge processing for the provisioning manifest" on page 161 for more information on the merge processing.

**product=<value>**

Specifies the product ID that will be used as the default by the platform. This results in the addition of the product property to the rcplauncher.properties file. This command is typically used by a branding feature that has provided a new product definition. The product attribute is removed during a feature disable operation.
**config.<configuration>.<id>=<value>**

Adds a config line to the rcplauncher.properties file. See the [Updating the rcplauncher.properties file](#) on page 89 for more information on the how the config attribute is used in the rcplauncher.properties file.

**file.copy <id>=<target><tab><source>**

Performs a file copy operation during the feature enablement operation.

- **<id>** Unique identifier differentiating multiple file copy commands
- **<target>** Target file
- **<tab>** Tab character that is used to separate source and target
- **<source>** Target file

When disabiling the feature, the <target> file is removed.

**product.name**

Specifies the registry key (Windows) or filename (Linux) that contains the path to the launcher program. See "Locating the Lotus Expeditor Client launcher program" on page 21 for more information on how the product.name command is used. The product.name command would typically be used as part of a branding operation. On feature disable, the registry key or file will be removed.

**osgi.splashPath=<value>**

Adds the osgi.splashPath property to the rcplauncher.properties file. The value is a URL referring to the location of the splash image. As an example, the branding feature for IBM Lotus Expeditor specifies the following:

```
osgi.splashPath=platform:/base/../rcp/eclipse/plugins/...com.ibm.rcp.platform.personality.branding
```

This command is typically used by a branding feature that needs to specify the splash image. On feature disable, the property will be removed from the rcplauncher.properties file.

**env.set.<id>=<value>**

Adds the env.set.<id>=<value> property to the rcpinstall.properties file. See "Updating the rcpinstall.properties file" on page 93 for more information on how the env.set property is used in the rcpinstall.properties file.

The following tokens are supported for paths in the handler.properties file:

- `product.install.dir`
- `rcp.home`
  
  The install directory for the platform.

- `features.dir`
  
  The features directory of the site to which the current feature is being installed.

- `old.feature.dir`
  
  The previous install directory of the feature if it exists.

- `plugin.dir`
  
  The plug-ins directory of the site to which the current feature is being installed.

- `plugin.dir.<id>`
  
  The install directory name for the given plug-in `<id>`.

- `old.plugin.dir.<id>`
  
  The previous install directory name for the given plug-in `<id>`.
The platform operating system (for example, win32).

The platform Architecture (for example, x86).

Creating instructions for desktop updates
The global install handler supports the creation of desktop-related objects.

Use the global install handler to:
- Create/delete a desktop link (Windows only)
- Create a program group (Windows and Linux only)
- Add/remove a link to the program group (Windows and Linux only)

To enable these functions, include a /links directory in your feature with properties files that fill in the following properties. It is also possible to include a directory (named for your operating system and architecture) that will read property files only when the proper operating system and architecture uses the feature. For example, if you create a directory named /win32x86 in your feature, the property files inside this directory would only be applied if your operating system is win32 and your architecture is x86.

The format of the properties files for creating links and program groups is as follows:

Table 19. Properties files format

<table>
<thead>
<tr>
<th>Link key name</th>
<th>String constants</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link.context</td>
<td>CONTEXT_DESKTOP</td>
<td>yes</td>
<td>Value can be one of the constants. Sets operation context to be either desktop or program group.</td>
</tr>
<tr>
<td></td>
<td>CONTEXT_PROGRAMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>link.foldername</td>
<td>N/A</td>
<td>no</td>
<td>This value is only used when the context of a link is CONTEXT_PROGRAMS. This is the program group folder name. If omitted, the link will be created in the root program group.</td>
</tr>
<tr>
<td>link.displayname</td>
<td>N/A</td>
<td>yes</td>
<td>This is the display name of the link. In windows the link filename would be &lt;display name&gt;.lnk.</td>
</tr>
<tr>
<td>link.iconpath</td>
<td>Can start with one of the following path constants:</td>
<td>no</td>
<td>Only required if the target exec does not have a built-in icon resource. Used to specify an icon for a program link. Example values:</td>
</tr>
<tr>
<td></td>
<td>$FEATURE_DIR</td>
<td></td>
<td>c:\winnt\icon.ico</td>
</tr>
<tr>
<td></td>
<td>$RCP_DIR</td>
<td></td>
<td>$FEATURE_DIR\icons\myicon.ico</td>
</tr>
<tr>
<td></td>
<td>$ECLIPSE_DIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$WORKSPACE_DIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>link.target</td>
<td>Can start with one of the following path constants:</td>
<td>yes</td>
<td>This is the path and filename of the target exec to launch. Example values:</td>
</tr>
<tr>
<td></td>
<td>$FEATURE_DIR</td>
<td></td>
<td>c:\winnt\notepad.exe</td>
</tr>
<tr>
<td></td>
<td>$RCP_DIR</td>
<td></td>
<td>$ECLIPSE_DIR\eclipse.exe</td>
</tr>
<tr>
<td></td>
<td>$ECLIPSE_DIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$WORKSPACE_DIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>link.args</td>
<td>N/A</td>
<td>no</td>
<td>This is the command line arguments for the target.</td>
</tr>
</tbody>
</table>
Table 19. Properties files format (continued)

<table>
<thead>
<tr>
<th>Link key name</th>
<th>String constants</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link.workingdir</td>
<td>Can start with one of the following path constants: $FEATURE_DIR $RCP_DIR $ECLIPSE_DIR $WORKSPACE_DIR</td>
<td>no</td>
<td>This is the path and filename of the target exec to launch. Example values: c:\myworkingdir $ECLIPSE_DIR</td>
</tr>
<tr>
<td>link.windowstate</td>
<td>Can be one of the following window state constants: WINDOW_STATE_NORMAL WINDOW_STATE_MIN WINDOW_STATE_MAX</td>
<td>no</td>
<td>This is the initial window state of the target exec.</td>
</tr>
</tbody>
</table>

Table 20. String constants - Link context constants

<table>
<thead>
<tr>
<th>String constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTEXT_DESKTOP</td>
<td>Indicates that this is a desktop link</td>
</tr>
<tr>
<td>CONTEXT_PROGRAMS</td>
<td>Indicates that this link is a program group.</td>
</tr>
</tbody>
</table>

Table 21. String constants - Path substitution constants

<table>
<thead>
<tr>
<th>String constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FEATURE_DIR</td>
<td>Used where allowed for path substitution. Resolves to the installed path to this feature's root directory. Example: C:\Documents and Settings\testuser\Application Data\Lotus\XPD\applications\eclipse\features\com.ibm.rcp.installhandler.test.feature</td>
</tr>
<tr>
<td>$RCP_DIR</td>
<td>Used where allowed for path substitution. Resolves to the root of the installed RCP client RCP directory. Example: c:\ibm\rich_client\rcp</td>
</tr>
<tr>
<td>$ECLIPSE_DIR</td>
<td>Used where allowed for path substitution. Resolves to the root of the installed RCP client Eclipse directory Example: c:\ibm\rich_client\eclipse</td>
</tr>
<tr>
<td>$WORKSPACE_DIR</td>
<td>Used where allowed for path substitution. Resolves to the active user's workspace root. Example: C:\Documents and Settings\someuser\Application Data\Lotus\XPD</td>
</tr>
</tbody>
</table>

Table 22. Link window state constants

<table>
<thead>
<tr>
<th>String constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WINDOW_STATE_NORMAL</td>
<td>Used by links to set the initial launch state of the target to normal.</td>
</tr>
<tr>
<td>WINDOW_STATE_MIN</td>
<td>Used by links to set the initial launch state of the target to minimized.</td>
</tr>
<tr>
<td>WINDOW_STATE_MAX</td>
<td>Used by links to set the initial launch state of the target to maximized.</td>
</tr>
</tbody>
</table>

Use the following tables to locate icons, depending on the type of user and features and the installation configuration.
• Windows desktop shortcut and menu locations:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Desktop shortcut location (single user configuration)</th>
<th>Menu location (single user configuration)</th>
<th>Desktop shortcut location (multi-user configuration)</th>
<th>Menu location (multi-user configuration)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin user and feature are shared</td>
<td>Shortcut goes to local desktop</td>
<td>Menu goes to local start menu</td>
<td>Shortcut goes to all users/desktop</td>
<td>Menu goes to All Users/Start Menu</td>
</tr>
<tr>
<td>Admin user and feature are not shared</td>
<td>Shortcut goes to local desktop</td>
<td>Menu goes to local start menu</td>
<td>Shortcut goes to admin/desktop</td>
<td>Menu goes to admin/Start Menu</td>
</tr>
<tr>
<td>Non-admin user and feature is shared</td>
<td>Shortcut goes to local desktop</td>
<td>Menu goes to local start menu</td>
<td>Shortcut goes to user/desktop</td>
<td>Menu goes to user/Start Menu</td>
</tr>
<tr>
<td>Non-admin user and feature is not shared</td>
<td>Shortcut goes to local desktop</td>
<td>Menu goes to local start menu</td>
<td>Shortcut goes to user/desktop</td>
<td>Menu goes to user/Start Menu</td>
</tr>
</tbody>
</table>

• Linux desktop shortcut and menu locations:

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Menu location (single user configuration)</th>
<th>Menu or shortcut location (multi-user configuration)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin user and feature are shared</td>
<td>Menu goes to /.local/share/applications</td>
<td>Menu goes to /usr/share/applications</td>
</tr>
<tr>
<td>Admin user and feature are not shared</td>
<td>Menu goes to /.local/share/applications</td>
<td>Menu goes to admin_home/.local/share/applications</td>
</tr>
<tr>
<td>Non-admin user and feature is shared</td>
<td>Menu goes to /.local/share/applications</td>
<td>Shortcut goes to user_home/.local/share/applications</td>
</tr>
<tr>
<td>Non-admin user and feature is not shared</td>
<td>Menu goes to /.local/share/applications</td>
<td>Shortcut goes to user_home/.local/share/applications</td>
</tr>
</tbody>
</table>

Adding the global install handler to a feature
You can add the global install handler to a feature.

To add the global install handler to the feature.xml, use the Feature Manifest Editor.
1. Open the feature.xml file using the Feature Manifest Editor.
2. Select the Installation tab
3. Enter com.ibm.rcp.installhandler.RCPInstallHandler as the Handler class. No entry is required in the Library attribute.
4. Select the Build tab in the Feature Manifest Editor.
5. Verify that the file handler:properties is checked in the binary build section.
6. Verify that the links directory, if used, is checked in the binary build section.
7. Save the feature.xml file.

Deploying features to the platform
This section describes the information and tasks necessary to deploy application and infrastructure features to the client platform.

Note: If you are using Windows Vista, be sure to review "Administering the platform on Windows Vista" on page 22 prior to changing the platform configuration.

There are four mechanisms for deploying features to the client platform:
Portal managed provisioning
A provisioning manifest is distributed from the Portal server to the client containing the
instructions on what features to install, and where to install them from. For more information, see
"Managing Rich Clients using the Portal server" on page 205.

Device Manager server-based provisioning
Specific Eclipse feature installation and management jobs are created at the server, distributed to
the client, and processed by the Enterprise Management Agent. For more information, see
"Managing using the Device Manager server" on page 181.

Using the provisioning application and provisioning manifest
The Provisioning Manifest is an XML file containing a set of instructions pertaining to the
features to install, enable, disable, or remove. This can be used by any user or as a mechanism
initiated from another management system. For more information, see "Managing using another
management system" on page 237.

Changes initiated using the provisioning application are specific to only a single runtime, but the
steps to request the changes can be easily replicated to many systems.

Using the application install and application management dialogs
The individual user specifically installs and manages the installed features using the available
dialogs. Changes are specific to only a single configuration, and the steps must be performed by
all users. This mode of operation is typically used by deployers in a test mode, by application
developers, or in scenarios where there might only be a few locally installed platforms.

Using any of the above methods, you can perform the following tasks.

Installing features in the multi-user configuration

Multiuser install is selected during the initial installation. As a result of selecting multi user install, the
install configuration property is set to multiuser.

As administrator, you are responsible for performing the initial installation, and should restrict directory
access for other users to the original install to read/execute access only.

When another user launches Lotus Expeditor Client for the first time, a new workspace will be created
based upon the installed platform using the provisioning manifest specified in the rcplauncher.properties
file.

On subsequent launches by the same user, the value of the provisioning.manifest.version in the user’s
cpinstall.properties file is compared to the value specified in the platforms rcplauncher.properties file. If
the values are different, then the launcher will start the provisioning application to reprocess the
provisioning manifest defined in the rcplauncher.properties file.

Installing applications for users in multi-user mode (non-Restricted Workbench)

As administrator, if you intend to install and enable features for every user on the system, then follow
these steps. Otherwise, see "Adding a new feature to the shared install sites" on page 159 and "Adding a
new feature to the user install site" on page 159.

1. If using Linux, modify owner permissions of the <users workspace> directory by specifying chown -R
   root <users workspace> where <users workspace> is the fully qualified directory containing logs,
   applications, and .config directories. Example: /home/myuser/Lotus/XPD
2. Launch the Lotus Expeditor Client with the -data <users workspace> command line parameters,
   where <users workspace> is the fully qualified directory containing logs, applications, and .config
directories. Windows Example: C:\Documents and Settings\myuser\Application Data\Lotus\XPD
   Linux Example: /home/myuser/Lotus/XPD
3. Click File → Application → Application Management, then locate the applications to enable for this user.
   For each feature that needs to be enabled, select the feature, then select the Enable action. If multiple features need to be enabled, you must enable them one at a time. Despite being prompted to restart after every action, you can enable multiple features before restarting.

4. Allow the Lotus Expeditor Client to restart when you are prompted and ensure the new applications are available.

5. Exit the Lotus Expeditor Client.

6. If using Linux, restore owner permissions of the <users workspace> directory by specifying chown -R "<username>"<users workspace> where <username> is the Linux login identity for the user.

   **Note:** These steps must be completed for each user you want enabled for the newly installed application.
   Additionally, it is possible for each user to enable the applications as needed by performing steps 2 and 3 above after launching the Lotus Expeditor Client normally.

### Installing applications for Restricted Workbench users

The Restricted Workbench prevents normal users from installing applications; requiring you, as the administrator, to perform all installations. To install applications for Restricted Workbench users, perform the following steps:

1. Log in to the operating system as the administrative user that installed the Lotus Expeditor Client.
2. Launch Lotus Expeditor Client.
3. Click File → Application → Install and use the installation wizard to install the desired applications.
4. Allow the Lotus Expeditor Client to restart when you are prompted.
5. Exit the Lotus Expeditor Client.

6. If using Linux, modify owner permissions of the <users workspace> directory by specifying chown -R root <users workspace> where <users workspace> is the fully qualified directory containing logs, applications, and .config directories. Example: /home/myuser/Lotus/XPD

7. Launch the Lotus Expeditor Client with the `data <users workspace>` command line parameters, where <users workspace> is the fully qualified directory containing logs, applications, and .config directories. Windows Example: C:\Documents and Settings\myuser\Application Data\Lotus\XPD
   Linux Example: /home/myuser/Lotus/XPD

8. Click File → Application → Application Management, then locate the applications to enable for this Restricted Workbench user.
   For each of the features that need to be enabled, select the feature, then select the Enable action. If multiple features need to be enabled, you must enable them one at a time. Despite being prompted to restart after every action, you can enable multiple features before restarting.

9. Allow the Lotus Expeditor Client to restart when you are prompted and ensure the new applications are available.

10. Exit the Lotus Expeditor Client.

11. If using Linux, restore owner permissions of the <users workspace> directory by specifying chown -R "<username>"<users workspace> where <username> is the Linux login identity for the user.

12. Log out of the operating system administrative user.

13. Log in to the operating system as the Restricted Workbench user. This will automatically start the Lotus Expeditor Client and the applications will now be available.

   **Note:** These steps must be completed for each Restricted Workbench user.
Adding a new feature to the shared install sites

The preferred model for an administrator to add a new feature to the platform in one of the shared sites (eclipse, rcp, or shared directories) is to do the following:

- Install the feature into one of the shared directories (by using the Device Manager server, Portal Managed provisioning, or other means)
- Update the provisioning manifest specified by the rcplauncher.properties file to include a definition of the feature that has been deployed
- Update the provisioning.manifest.version in the rcplauncher.properties to a new value

Since the provisioning.manifest.version will not match the values in the users workspaces, as they individually start their platform, they will automatically receive the new feature enabled in their configuration.

The administrator can also manually add a new feature onto the platform in one of the shared install sites (eclipse, rcp, or shared directories). The feature will be installed following the install site selection rules defined in [“Controlling the feature install location” on page 148]. Once the feature has been added to the platform, each user must individually enable that feature in their configuration.

Adding a new feature to the user install site

Users are permitted to install features into an install site present in their workspace. These features are available only to the user that has installed the feature.

Additionally, if the deployer wants applications to be individually deployed to users’ workspaces, the deployer can also follow similar steps as Adding a new feature to the shared install site. By updating the provisioning manifest to include the feature, but specify an attribute shared with a value false, the feature is directed to install into the user site.

Deleting features from the shared install sites

The users need to remove features from their workspaces prior to the administrator removing features from the platform. If the feature is deleted from the platform prior to being removed from the user workspaces, then the users will encounter multiple errors regarding missing features and will not be able to effect any changes to the platform.

The preferred model for an administrator to effect a removal of a feature from the platform is the following:

- Update the provisioning manifest specified by the rcplauncher.properties file to change the action for the feature from install to disable, or add the feature to the provisioning manifest with an action of disable
- Update the provisioning.manifest.version in the rcplauncher.properties to a new value

As users launch their platform, the feature will be disabled from their configuration. When the feature has been removed from all user configurations, the administrator can then remove the feature from the shared install site.

Deleting features from the user install sites

Users can delete features from their own configuration, but cannot affect the features installed in the shared site or in other user sites.

If the deployer had configured the provisioning manifest to install the feature into the user install site, then the action for the feature can be changed to “uninstall” to remove it from each of the users configurations when they restart their platform.
Using the provisioning manifest

The Expeditor Client installation process consists of two steps: installation of core platform files and provisioning of additional components. Depending on the desired install configuration (user or multiuser), differing sets of components are provisioned to the product from update sites.

6.1.1 The set of components to be provisioned is defined within provisioning manifests, 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 provisioning manifests must be named user.xml or multiuser.xml (corresponding to the available install configurations).

6.1.2 The set of components to be provisioned is defined within a provisioning manifest, which is processed by the installer and platform provisioning system. The file must be named install.xml and be located in a deploy/directory located in the same location as the installer executable.

The Expeditor Client CD includes a provisioning manifest for each of these install configurations. 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 install configurations include a customized set of components.

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

6.1.1 During the installation process, the provisioning manifest corresponding to the chosen install configuration is processed (replacing any supported variables) and copied to \</product
destination>/rcp/deploy/install.xml.

6.1.2 During the installation process, the provisioning manifest is processed (replacing any supported variables) and copied to \</product
destination>/rcp/deploy/install.xml.

The location of this file is then passed to the platform provisioning component to begin the provisioning process.

The Expeditor platform defines a default set of features that will be provisioned during the initial installation. The list of features can be updated prior to installation to either remove features that are not desired, or to add custom or application specific features.

In all configurations, the launcher (rcplauncher.exe) reads the rcplauncher.properties to determine the provisioning.manifest.version property, and compares to the version currently specified in the rcpinstall.properties. If the version in rcplauncher.properties is greater than the version in the rcpinstall.properties file, then launcher will start the platform provisioning application to update the platform based upon the provisioning manifest (specified in the rcplauncher.properties).

In the multiuser platform configuration, the provisioning manifest is therefore used as the basis for creating the new workspace for every user.

By updating the provisioning manifest contents, and the provisioning.manifest.version, the launcher will start the provisioning application to process the provisioning manifest.

While the provisioning manifest can be directly updated on the system, the preferred model for distributing a new manifest is to use the mergemanifest command provided by the Global Install Handler. The install handler provides a mechanism to deploy a new provisioning manifest and to merge it with the existing manifest present on the system. This enables the deployment of a subset of new changes that will be merged into the existing manifest without requiring the specification of an entire file.
The mergemanifest command is also used to deliver a new version of the provisioning manifest, such as occurs during an upgrade of the existing platform.

**Understanding the merge processing for the provisioning manifest**

The provisioning manifest generally serves two purposes. During the initial installation, the provisioning manifest provides instructions for the install program to specify the options available for installation, and also specifies the features that need to be installed during the initial provisioning. Subsequent to the install, the provisioning manifest is used during the construction of new workspaces to provide the initial configuration for those workspaces.

The provisioning manifest is also used during subsequent provisioning operations to make modifications to the installed platform. During upgrades initiated through the install program, or upgrades initiated via an install update site, a new provisioning manifest may be provided to the platform to provide instructions for the new workspace configuration. Since the original provisioning manifest may have been modified by a deployer to customize the platform, the new provisioning manifest is merged with the current provisioning manifest.

When a new provisioning manifest is merged with an existing provisioning manifest, the new provisioning manifest can contain mergeaction attributes on the features and installfeatures present in the provisioning manifest. The mergeaction attributes provide the rules on how to merge the content of the two provisioning manifest files.

The general principle is that what was previously installed will stay installed. Generally only upgrades to existing components will be performed, and new components will not be installed, unless required as a dependency.

More specifically, the following rules control the merge process:

**Table 25. Merge process control rules**

<table>
<thead>
<tr>
<th>Current manifest</th>
<th>New Manifest</th>
<th>Result if Merge=&quot;Add&quot;</th>
<th>Result if Merge=&quot;Remove&quot;</th>
<th>Result if Merge not specified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature specified</td>
<td>Feature specified</td>
<td>n/a</td>
<td>Feature removed from the provisioning manifest</td>
<td>Feature retained with new feature attributes</td>
</tr>
<tr>
<td>Feature specified</td>
<td>Feature not specified</td>
<td>n/a</td>
<td>n/a</td>
<td>Feature retained with old feature attributes</td>
</tr>
<tr>
<td>Feature not specified</td>
<td>Feature specified</td>
<td>Feature retained with new feature attributes</td>
<td>Feature removed from the provisioning manifest</td>
<td>Feature removed from the provisioning manifest</td>
</tr>
</tbody>
</table>

The merge action specified here is defined in the provisioning manifest definition. The merge action only applies to the provisioning manifest and does not apply to the features already present on the file system. During the merge process, the merge action attribute is removed.

In the provisioning manifest, features are elements of the installfeature element. The table above defines the results on a feature by feature basis within an install feature with the same ID.

Install features also define merge actions that are processed according to this table:

**Table 26. Additional merge actions**

<table>
<thead>
<tr>
<th>Current manifest</th>
<th>New Manifest</th>
<th>Result with merge=&quot;Add&quot;</th>
<th>Result with merge=&quot;Remove&quot;</th>
<th>Result with no merge attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install feature specified</td>
<td>Install feature specified</td>
<td>n/a</td>
<td>Entire install feature is removed from manifest</td>
<td>Features within the install feature are individually merged</td>
</tr>
</tbody>
</table>
The following scenarios will help illustrate the processing during the merge operation.

**Scenario I**

The product’s original manifest was used during install, and the product installer is used with the upgrade product’s new manifest.

The original manifest contained these definitions:

```xml
<feature id="A" version="1.0"/>
<feature id="B" version="1.0"/>
<feature id="C" version="1.0"/>
```

The new manifest contained these definitions:

```xml
<feature id="A" version="1.0.1"/>
<feature id="B" version="1.2"/>
<feature id="C" version="2.0"/>
```

The features already installed on the system will be upgraded as part of the new installation. New optional components provided as part of the product will not be added to the manifest since they were not previously present. New required components will be specified in the manifest with `merge="add"` attributes. The resulting manifest will be the following:

```xml
<feature id="A" version="1.0.1"/>
<feature id="B" version="1.2"/>
<feature id="C" version="2.0"/>
```

**Scenario II**

The product’s original manifest was modified by the deployer prior to initial install to remove some of the product features. The product installer is used with the upgrade product’s new manifest (without modifications).

The original manifest contained these definitions:

```xml
<feature id="A" version="1.0"/>
<feature id="C" version="1.0"/>
```

The new manifest contained these definitions:

```xml
<feature id="A" version="1.0.1"/>
<feature id="B" version="1.2"/>
<feature id="C" version="2.0"/>
```

Only the features already present in the original manifest will be in the resultant manifest. Original product features removed by the deployer will be removed from the resultant manifest, and therefore will not be installed or upgraded. New optional components provided as part of the product will not be added to the manifest since they were not previously present. New required components will be retained in the manifest since they were specified with the `merge="add"` attributes.

The resulting manifest will be the following:
Scenario III

The product's original manifest was modified by the deployer prior to initial install to add additional custom features. The product installer is used with the upgrade product's new manifest.

The original manifest contained these definitions:
<feature id="A" version="1.0"/>
<feature id="B" version="1.0"/>
<feature id="C" version="1.0"/>
<feature id="D" version="4.0"/>

The new manifest contained these definitions:
<feature id="A" version="1.0.1"/>
<feature id="B" version="1.2"/>
<feature id="C" version="2.0"/>
<feature id="D" version="4.0"/>

All of the features present in the original manifest will be retained, including the custom features. Upgrades to the product components will be retained since they are present in the existing manifest. New optional components will not be added to the manifest. New required components will be added to the manifest since they contain the merge="add" attribute.

The resulting manifest will contain the following:
<feature id="A" version="1.0.1"/>
<feature id="B" version="1.2"/>
<feature id="C" version="2.0"/>
<feature id="D" version="4.0"/>

Scenario IV

The deployer needs to remove a feature from the resulting manifest. The feature must be specified in the upgrade manifest with a mergeaction="remove".

The original manifest contained these definitions:
<feature id="A" version="1.0"/>
<feature id="B" version="1.0"/>
<feature id="C" version="1.0"/>

The new manifest contained these definitions:
<feature id="A" version="1.0"/>
<feature id="B" version="1.2" mergeaction="remove"/>
<feature id="C" version="2.0"/>

The resulting manifest will contain the following definitions:
<feature id="A" version="1.0.1"/>
<feature id="C" version="2.0"/>

Scenario V

The deployer needs to add a new feature to the system via the merge process. The feature must be specified in the upgrade manifest with a mergeaction = add.

The original manifest contained these definitions:
<feature id="A" version="1.0"/>
<feature id="B" version="1.0"/>
<feature id="C" version="1.0"/>
The new manifest contained these definitions:

```xml
<feature id="A" version="1.0.1"/>
<feature id="B" version="1.2"/>
<feature id="C" version="2.0"/>
<feature id="E" version="1.2" mergeaction="add"/>
```

The resulting manifest will contain the following definitions:

```xml
<feature id="A" version="1.0.1"/>
<feature id="B" version="1.2"/>
<feature id="C" version="2.0"/>
<feature id="E" version="1.2"/>
```

**Scenario VI**

A new installfeature is available containing optional features, and needs to be added to the provisioning manifest. The original provisioning manifest contains the following:

```xml
<installfeature id="A" name="Required Feature A" required="true">
  ...
</installfeature>
<installfeature id="B" name="Optional Feature B" required="false">
  ...
</installfeature>
```

The upgrade provisioning manifest contains the following:

```xml
<installfeature id="A" name="Required Feature A" required="true">
  ...
</installfeature>
<installfeature id="B" name="Optional Feature B" required="false">
  ...
</installfeature>
<installfeature id="C" name="Optional Feature C" required="false" mergeaction="add">
  ...
</installfeature>
```

Following the rules defined above, since neither installfeature A nor B includes a mergeaction, the features present in each of these installfeatures will be merged together. Since installfeature C contains a mergeaction specified as add, then installfeature C will be retained in the resulting manifest.

```xml
<installfeature id="A" name="Required Feature A" required="true">
  ...
</installfeature>
<installfeature id="B" name="Required Feature B" required="false">
  ...
</installfeature>
<installfeature id="C" name="Optional Feature C" required="false">
  ...
</installfeature>
```

**Scenario VII**

One of the features has been enhanced, but has now become an optional installation component. The original provisioning manifest contains the following:

```xml
<installfeature id="A" name="Required Feature A" required="true">
  <feature id="A.1" version="1.0"/>
  ...
</installfeature>
<installfeature id="B" name="Optional Feature B" required="false">
  ...
</installfeature>
<installfeature id="C" name="Optional Feature C" required="false">
  ...
</installfeature>
```

Lotus Expeditor: Assembling and Deploying Lotus Expeditor Applications
The upgrade provisioning manifest contains the following:

```xml
<installfeature id="A" name="Required Feature A" required="true">
  <feature id="A.1" version="1.0" mergeaction="remove"/>
...
</installfeature>
<installfeature id="B" name="Optional Feature B" required="false">
...
</installfeature>
<installfeature id="C" name="Optional Feature C" required="false" mergeaction="add">
  <feature id="A.1" version="2.0"/>
...
</installfeature>
```

Following the rules defined above, since neither installfeature A nor B includes a mergeaction, the features present in each of these installfeatures will be merged together. Since feature A.1 has a mergeaction of remove from installfeature A, it will be removed from installfeature A in the result. Since installfeature C contains a mergeaction specified as add, then installfeature C will be retained in the resulting manifest. All features present in installfeature C will be added, so feature A.1, previously removed from installfeature A, is now contained in installfeature C.

```xml
<installfeature id="A" name="Required Feature A" required="true">
...
</installfeature>
<installfeature id="B" name="Optional Feature B" required="false">
...
</installfeature>
<installfeature id="C" name="Optional Feature C" required="false">
  <feature id="A.1" version="2.0"/>
...
</installfeature>
```

### Provisioning manifest DTD

The provisioning manifest is an XML file that lists the features to be installed during the provisioning portion of the install process.

The structure of the provisioning manifest is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ibm-portal-composite SYSTEM "com.ibm.rcp.installmanifest.dtd">
<!ELEMENT ibm-portal-composite (domain-object)>*
<!ELEMENT domain-object (object-data)>*
<!ATTLIST domain-object name CDATA #FIXED "com.ibm.rcp.installmanifest">
```

Following the rules defined above, since neither installfeature A nor B includes a mergeaction, the features present in each of these installfeatures will be merged together. Since feature A.1 has a mergeaction of remove from installfeature A, it will be removed from installfeature A in the result. Since installfeature C contains a mergeaction specified as add, then installfeature C will be retained in the resulting manifest. All features present in installfeature C will be added, so feature A.1, previously removed from installfeature A, is now contained in installfeature C.

```xml
<installfeature id="A" name="Required Feature A" required="true">
...
</installfeature>
<installfeature id="B" name="Optional Feature B" required="false">
...
</installfeature>
<installfeature id="C" name="Optional Feature C" required="false">
  <feature id="A.1" version="2.0"/>
...
</installfeature>
```

The provisioning manifest is an XML file that lists the features to be installed during the provisioning portion of the install process.

The structure of the provisioning manifest is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ibm-portal-composite SYSTEM "com.ibm.rcp.installmanifest.dtd">
<!ELEMENT ibm-portal-composite (domain-object)>*
<!ELEMENT domain-object (object-data)>*
<!ATTLIST domain-object name CDATA #FIXED "com.ibm.rcp.installmanifest">
```

Following the rules defined above, since neither installfeature A nor B includes a mergeaction, the features present in each of these installfeatures will be merged together. Since feature A.1 has a mergeaction of remove from installfeature A, it will be removed from installfeature A in the result. Since installfeature C contains a mergeaction specified as add, then installfeature C will be retained in the resulting manifest. All features present in installfeature C will be added, so feature A.1, previously removed from installfeature A, is now contained in installfeature C.

```xml
<installfeature id="A" name="Required Feature A" required="true">
...
</installfeature>
<installfeature id="B" name="Optional Feature B" required="false">
...
</installfeature>
<installfeature id="C" name="Optional Feature C" required="false">
  <feature id="A.1" version="2.0"/>
...
</installfeature>
```
UI when the ReadDeploymentManifest bean is used in the offering installer.

restartPersonality (optional)
Used to indicate the personality with which the platform should be restarted once the provisioning of all the features has succeeded.

required (optional)
Denotes whether this is a required installfeature or can be optionally installed. Default value is false.

mergeaction (optional)
Selected values are "add" and "remove"

During upgrade scenarios, a pre-existing provisioning manifest will be merged with a new manifest, according to the following rules.

The mergeaction attribute is processed only from the new provisioning manifest, and is ignored in the old manifest.

add
forces an installfeature specified in the new provisioning manifest file to be retained in the merged result.

remove
forces an installfeature specified in the new provisioning manifest file to be removed from the merged result.

If the mergeaction attribute is not present, then the features within the installfeature are processed against the install feature of the identical id

-->

<!ELEMENT installfeature (requirements)>
<!ATTLIST installfeature
  id CDATA #REQUIRED
  name CDATA #IMPLIED
  restartPersonality CDATA #IMPLIED
  required CDATA #IMPLIED
  mergeaction (add|remove) #IMPLIED>

<!--The requirements tag denotes a requirements block which will contribute a list of required features to be used by the provisioning system. -->
<!ELEMENT requirements (feature)>+

<!-- This tag denotes a feature that is required by the platform for a given deployment.

This tag will have the following attributes:

  id
The id of the feature to be provisioned

  version
The version of the feature to be provisioned

  match(optional)
Used to indicate the desired match rule.

According to the manifest definition in the Eclipse help, the match rules are as follows:
perfect
Dependent plug-in version must match exactly the specified version. If "patch" is "true", "perfect" is assumed and other values cannot be set.

equivalent
Dependent plug-in version must be at least at the version specified, or at a higher service level (major and minor version levels must equal the specified version).

compatible
Dependent plug-in version must be at least at the version specified, or at a higher service level or minor level (major version level must equal the specified version).

greaterOrEqual
Dependent plug-in version must be at least at the version specified, or at a higher service, minor or major level.

Match attributes are specified in both the provisioning manifest (user.xml or multiuser.xml) and in the feature itself.

Match attributes are specified in both the provisioning manifest (install.xml) and in the feature itself.

The match attributes specified inside the feature apply to the feature requirements and plugin requirements for the feature. These are processed by the update manager to determine whether all requirements for the feature have been satisfied.

The match attributes specified in the provisioning manifest are used as a predetermination for installing the feature.

If not specified, the setting of the Eclipse preference org.eclipse.update.core/org.eclipse.update.core.updateVersions is used.

url (optional)
The URL to the update site where the feature can be found. An update site is any valid file URL which contains a site manifest file, a features folder, and a plugins folder.
Example: url=http://www.myupdatehost.com/updateSite
Example: url=file:/C:/Program Files/application/updateSite

"${installer.root}" is an optional token that is replaced by the installer when the manifest is processed prior to initial provisioning. The parameter is replaced with the directory from which the installer has launched, and can therefore only be used in conjunction with a relative update site path.
Example: "${installer.root}" = "file:/D:/ProductCD/install/"
url="${installer.root}/../updates" translates to file:/D:/ProductCD/../updates/

If a URL is not specified, then it is expected that the com.ibm.rcp.provisioning/updateSiteList preference has been updated with a list of update sites from which features will be installed. During the installation process, the installer automatically provides an update site that is used if the feature does not otherwise have a url specified. For more information, see: Configuring feature search order on page 50

size(optional)
The installed size (in kilobytes) of the feature. It is used by the install application to calculate the total size of the installed product. The size attribute is currently used only by the install application.

download-size(optional)
The download size (in kilobytes) of the feature. It is used by the install application to calculate the amount of temporary space required during the install processing. The download-size attribute is currently used only by the install application.

shared(optional)
If set to true, the feature will be installed into the shared site (if no other feature version or affinity is satisfied). If false, the feature will be installed into the user site (if no other feature version or affinity is satisfied). The default setting will depend on the install configuration. If a multiuser install configuration is being used, the default is true. If a single user install configuration is being used, the default is false.

remove(optional)
Remove this feature. This attribute is deprecated. Specify action="uninstall" instead.

action(optional)
Selected values are as follows:

install
Install and configure the feature (default)
installonly
Install, but do not configure the feature
enable
Enable a feature that is currently installed
disable
Disable a feature that is currently installed
uninstall
Disable, then uninstall a feature
ignore
Perform no actions on this feature

mergeaction(optional)
Selected values are "add" and "remove"

During upgrade scenarios, a pre-existing provisioning manifest will be merged with a new manifest, according to the following rules.

The mergeaction attribute is processed only from the new provisioning manifest, and is ignored in the old manifest.

add
forces a feature specified in the new provisioning manifest file to be retained in the merged result.

remove
forces a feature specified in the new install manifest file to be removed from the merged result.
If the feature is present in the old manifest, and the feature is present in the new manifest

mergeaction attribute not specified
The feature is retained with the attributes defined in the new manifest

mergeaction = "add"
new feature is retained completely

mergeaction = "remove"
feature is removed

If the installfeature is present in the old manifest, and does not exist in the new manifest, it will be retained completely

If the installfeature is not present in the old manifest, it will only exist in the result if mergeaction="add" is specified

-->

<?ELEMENT feature EMPTY>
<?ATTLIST feature
 id CDATA #REQUIRED
 url CDATA #IMPLIED
 version CDATA #REQUIRED
 match (perfect|compatible|equivalent|greaterOrEqual) #IMPLIED
 size CDATA #IMPLIED
 download-size CDATA #IMPLIED
 shared CDATA #IMPLIED
 mergeaction (add|remove) #IMPLIED
 action (install|installonly|enable|disable|uninstall|ignore)
   "install"
 remove CDATA "true" >

Using this provisioning manifest structure, any number of features can be specified, and each feature can come from different originating update sites. Each feature can provide a match rule for determining if it should be installed or updated. The feature’s destination site and configuration can also be specified (using the shared and config attributes), and provided sizes will be added to the size of the install reported by the installer so that an accurate total platform size (including installed and provisioned files) can be presented to the user.

Examples
1. Installation of the Order Entry sample located on an Lotus Expeditor Client CD-ROM:

   Note: The feature version shown here may differ from the version shipped with the product.

   <ibm-portal-composite>
   <domain-object name="com.ibm.rcp.installmanifest">
   <object-data>
   <install>
   <installfeature>
   <requirements>
   <feature id="com.ibm.rcp.samples.orderentry.feature" version="6.1.0.0-20060815" action="install" size="25"/>
   </requirements>
   </installfeature>
   </install>
   </object-data>
   </domain-object>
   </ibm-portal-composite>

2. Installation of features located on two different HTTP server sites:

   <ibm-portal-composite>
   <domain-object name="com.ibm.rcp.installmanifest">
   <object-data>
   <install>
   <installfeature>
   <requirements>
   <feature id="com.aaa.feature1" version="1.0.0" url="http://company1.com/updates" action="install" size="900"/>
   <feature id="com.aaa.feature2" version="5.6.0" url="http://company1.com/updates" action="install" size="300"/>
   <feature id="com.bbb.feature1" version="1.2.0" url="http://company2.com/site" action="install" size="50" shared="false"/>
   <feature id="com.bbb.feature2" version="3.0.5" url="http://company2.com/site action="install" size="2100" match="equivalent"/>
   </requirements>
   </installfeature>
   </install>
   </object-data>
   </domain-object>
   </ibm-portal-composite>
Installing additional languages content
The IBM Lotus Expeditor runtimes have been translated for Group 1, Group 2, and Group 3 languages.

The Group 1 languages include:
• Chinese (simplified)
• Chinese (traditional)
• Japanese
• Korean
• French
• German
• Italian
• Spanish
• Portuguese (Brazilian)

The Group 2 languages include:
• Arabic
• Czech
• Danish
• Dutch
• Finnish
• Greek
• Hebrew
• Hungarian
• Norwegian
• Polish
• Portuguese
• Russian
• Swedish
• Turkish

Note: When installing a platform on Linux, Hebrew and Arabic locale specific fragments are not installed on Linux. Installation of the Hebrew and Arabic locales on Linux and launching with those locales is also not supported.

The Group 3 languages include:
• Thai
• Catalan
• Romanian
• Slovak
• Slovenian
• Ukrainian

U.S. English is packaged within all IBM Lotus Expeditor features and plug-ins and is available by default.
During the initial installation of the platform, only the national language features that contain the current locale are installed and enabled.

**Note:** If you are using Windows Vista, be sure to review the link about “Administering the platform on Windows Vista” on page 22 prior to installing the language features.

To enable locale specific content, additional national language features must be installed. The platform must be started in the desired locale to install and enable the national language features for that locale. This can be accomplished by switching the system to the appropriate locale before starting IBM Lotus Expeditor. It can also be accomplished by starting IBM Lotus Expeditor with a --nl command argument. For example, to start the platform in French, enter `<install dir>/rcp/rcplauncher.exe --nl fr`.

When IBM Lotus Expeditor has started, install the appropriate group of languages. If you need to install another group, you must shutdown IBM Lotus Expeditor and change locale using one of the above methods.

If National Language Features are installed but do not match the current platform locale, then the features will not be enabled.

When viewed in the **Application > Application Management** dialog, the list of installed and enabled features is filtered based upon the locale. For example, if both the Group 1 and Group 2 language features are installed, and you have started the platform using the Finnish locale, then the Application Management dialog will only show the Group 2 language features, and will not show the Group 1 language features.

When supporting a language other than English on a multi-user installation, you, as the administrator, must install the correct groups of languages. If space permits, install all language groups. If you only need French then install only the Group 1 language translation. There are restrictions on how this can be done. To install Group 1 languages, you must start the platform in a language that Group 1 supports.

The provisioning system will automatically enable a single locale group per user and does not support automatic switching between locale groups. In a multi-user environment, each user can have a different locale group as long as the administrator has installed all of the required NL packs before users launch IBM Lotus Expeditor and provision their own workspaces.

**Upgrading features on the platform**

Choose from various mechanisms for upgrading features on the platform.

You can upgrade the features existing on the platform using these mechanisms:

- Use the Update Manager user interface (File > Application > Install > Search for new features to install) to manually install new versions of features
- Distribute new feature versions through the Composite Application Infrastructure (CAI) and provisioning from the server.
- Distribute new feature versions through the Device Manager server.
- Use an alternative management system to provide a provisioning manifest and call the ProvisioningApplication to install new updates.
- Configure the platform to use an automatically scheduled Scan for Updates

The Scan for Updates scans the set of update sites defined by features on the platform and the policy, as filtered by the white list processing. It looks for updates to existing features on the platform. The set of features located will be proposed as updates to the platform. The scan for updates will not propose for installation new features that are not previously present on the system.
For any of the processes outlined above, the updated features will be installed and enabled in the current configuration. According to the setting for the `com.ibm.rcp.provisioning/feature.history.size` preference, once a feature with an updated version has been installed and enabled, previous versions can be removed.

Unless the platform provisioning manifest (defined in the `rcplauncher.properties` and defaulting to the `rcp\deploy\install.xml` file) is updated to reflect these feature updates, new workspace creation will attempt to use the previously defined feature versions. If these previous feature versions are no longer installed, new workspace creation might fail.

The mergemanifest command of the platform global install handler provides a mechanism for automatically updating the provisioning manifest to stay consistent with the installed features and provides a powerful mechanism for updating the platform. Using the mergemanifest reduces the focus from updating several features, to focusing on the update of one feature, which will then drive the remainder of the feature updates. The following scenario illustrates how the mergemanifest command for the platform global install handler can be used:

1. Define a feature called `CompanyApplications` and give it a version 1.0.0. This feature does not need to contain any plug-ins. Distribute the `CompanyApplications` feature on an update site.
2. Create an update policy XML file on a Web or shared file server that refers all scan for updates to the update site containing only the `CompanyApplications` feature.
3. Customize the initial platform installation to include the `CompanyApplications` feature.
4. Customize the `plugin_customization.ini` file to include the update policy URL preference and configure the platforms to automatically scan for updates.
5. Install the platform, which will now install the `CompanyApplications` feature. The provisioning manifest will contain the `CompanyApplications` feature, and, if a new workspace is required to be created, the `CompanyApplications` feature will be in all platforms.

When there are updates to be applied to the platform, perform the following steps:

1. Create a provisioning manifest file that references the new or updated features that need to be distributed to the clients and containing the appropriate `mergeaction` tags. Include the provisioning manifest file in the feature.
2. Create a handler.properties file to contain the mergemanifest command referencing the provisioning manifest included in the feature.
3. Update the `CompanyApplications` feature to a new version, such as 1.0.1.
4. Distribute the other new or updated features to a second update site. (By using a second update site, the scan for updates will only identify the `CompanyApplications` feature to be updated, reducing the confusion in the number of features to be distributed.)
5. Distribute the updated `CompanyApplications` feature to the update site referenced by the update policy.

As the clients perform their scheduled scan for updates, they will detect the new `CompanyApplications` feature and propose the installation of the feature. When the feature is installed and enabled, the mergemanifest command will be processed, which results in an update to the provisioning manifest and an update to the `provisioning.manifest.version` property in the `rcplauncher.properties` file. On restart of the platform, the platform launcher will detect the difference in the `provisioning.manifest.version` from its previous setting (compared to the value in the `rcpinstall.properties` file) and process the contents of the provisioning manifest file. Since new or updated features were merged into the provisioning manifest by the mergemanifest command, the `ProvisioningApplication` will process those changes, installing or enabling the new or updated features for each workspace.

While this scenario uses the scan for updates, the same basic process can be repeated by distributing the `CompanyApplications` feature in other ways, such as through the Device Manager server or through the Portal server.
6.1.1 The IBM Lotus Expeditor platform branding feature contains a mergemanifest command for its install handler, referring to a user.xml contained within the branding feature. If the IBM Lotus Expeditor branding feature is installed from an update site, it will merge the contents of the user.xml with the provisioning manifest on the platform.

6.1.2 The IBM Lotus Expeditor platform branding feature contains a mergemanifest command for its install handler, referring to a install.xml contained within the branding feature. If the IBM Lotus Expeditor branding feature is installed from an update site, it will merge the contents of the install.xml with the provisioning manifest on the platform.

As the IBM Lotus Expeditor provides future platform upgrades by updating the branding feature on the platform, other updates will automatically be installed using the provisioning manifest.

Uninstalling features on the platform
When new versions of existing features are installed on the platform, the previous versions are immediately disabled.

Depending on the preference setting for com.ibm.rcp.provisioning.feature.history.size, the previous versions may also be physically removed from the system.

Features that were installed into one workspace cannot be uninstalled from a different workspace. For example, when you initially launch IBM Lotus Expeditor using the default workspace, a set of features are installed and provisioned for you. If you later launch with a different workspace using the rcpData launch option, the features that were installed into the default workspace cannot be uninstalled from the new workspace. If you attempt to uninstall these features, they will not be uninstalled and errors will be written to the error log.

Contributing to the deferred actions list
The provisioning system maintains a set of requests that are deferred for processing until the platform restarts.

Physical uninstall of features is deferred until the platform restarts. Actions can only be added to the list as an indirect result of requesting the uninstall of a feature.

Deferred actions are processed through the org.eclipse.ui.startup extension point. Therefore, the deferred actions will run only when the workbench is available, which is after the user authentication has occurred. The Provisioning Application will synchronously process deferred actions prior to beginning the requested operations.

Deploying applications through Web Start
This section describes how to export and deploy a Client Services RCP application through the Java Web Start mechanism.

The Java Web Start mechanism supports the deployment of Java applications from a Web server, onto a client. The Client Services Web Start tools support the packaging of RCP applications for subsequent deployment through the Web Start mechanism. These applications run against an installed IBM Lotus Expeditor runtime on the client. Each application is launched as a self-contained RCP application running in its own VM. The application is not installed into the IBM Lotus Expeditor platform. Instead, it can reference required features from the IBM Lotus Expeditor platform that it needs. When the application ends, it may be cached on the client for future invocations, or may be removed if the cache is full. In this respect, the application is transient, like Java applications that are run through Web Start.

Further information on Java Web Start can be found in the documentation provided with Java J2SE 1.5.
Supported applications and environments
All application types that can be run on the Lotus Expeditor Client can also be deployed and run through Web Start. All that is required is that the application be packaged in one or more features, and that a product configuration be created for it.

Java Web Start requires J2SE support on the client. For this reason, only the J2SE 1.5 JRE that is delivered through the DRE is supported as a Java runtime. The jclDesktop and jclDevice JREs are not supported.

Supported operating system platforms include:
• Microsoft Windows XP Professional Service Pack 1 and 2
• Microsoft Windows XP Home Edition Service Pack 1 and 2
• Microsoft Windows Vista
• Red Hat EL 4.0 WS with GTK support - Update 4
• Red Hat EL 5.0 WS with GTK support
• SUSE Linux Enterprise Desktop 10

J2SE 1.5 must be installed on the client. For more information, see “Installing Web Start support on the client” on page 176.

Differences from the Eclipse Web Start support
This section describes the differences between the Client Services Web Start support and the Eclipse Web Start support.

The Client Services Web Start tools differ from the Eclipse tools in the following areas:
• The Eclipse Web Start tools package up an RCP application with all of the features referenced by the application. In essence, this provides a standalone application. The Client Services tools assume the existence of an installed IBM Lotus Expeditor runtime on the client machine. This allows the RCP application’s packaging to only contain the application unique features. The benefits of this are two-fold:
  – The application to be downloaded from the server is potentially much smaller, leading to a faster download.
  – The IBM Lotus Expeditor runtime can be updated independently from the application, avoiding the need to re-build and re-deploy all applications in order to take advantage of IBM Lotus Expeditor feature updates.
• The IBM Lotus Expeditor tools understand the IBM Lotus Expeditor runtime, and can properly set up the application’s IBM Lotus Expeditor launch environment.
• Only the common launch jar that is provided with the Client Services Web Start support needs to be signed. This can be done once by the administrator since this jar is common to all Client Services Web Start applications.

Multi-user support
A Web Start-deployed application references the shared features from a multi-user installation.

Web Start does not reference a particular user’s IBM Lotus Expeditor configuration.

Structure of a Client Services Web Start application
Expeditor Deployers: Structure of a Client Services Web Start application

The RCP JNLP application contains the following resources:
Common launch component
Contains common logic required to setup and launch the RCP application. It is contained in jar file com.ibm.rcp.jnlp.launch_1.0.0.jar.

Top level JNLP descriptor
Specifies the common launch component and the application extension.

Application extension JNLP descriptor
Specifies the application extension.

Application jar
Contains the RCP application itself in the form of an Eclipse install directory.

Splitting the JNLP description into a top level jnlp and an extension jnlp allows full access security to be specified at the top level for the common launch component, without requiring the resources referenced in the application extension to be signed. Only the common launch component com.ibm.rcp.jnlp.launch_1.0.0.jar needs to be signed, and a single instance can be shared by all of the RCP applications.

Developing a Web Start application
This section describes the basics involved in developing a Web Start application.

All Client Services application types can be deployed through Web Start. The application itself is a Client Services application, and as such, is developed using the existing tools available in the IBM Lotus Expeditor Toolkit. It can be tested using the Client Services launch or Client Services server support.

The Client Services Web Start export wizard assembles the necessary metadata and packages the application in a form suitable for Web Start deployment. See “Exporting the Web Start application” for more information on this process.

Exporting the Web Start application
This section describes how to export the Web Start application.

The following prerequisites must be met before an application can be exported for Web Start:
• The application must be structured as one or more features.
• A product configuration must be created which defines the features that make up the application along with application launch details.

Creating and editing a product configuration follows the same steps as used when exporting a client runtime image. See “Creating a Platform Builder configuration” on page 141 for further information.

When one or more features have been created for the application and a product configuration has been created, the application can be exported using the following steps:
1. Select File → Export.
2. From the Client Services folder, select Client Services Web Start, and click Next.
3. Select the product configuration for the application.
4. Specify the destination directory for the export, and optionally change the application jar name.
5. Optionally, provide a version specification for the target runtime. Blank (default) indicates any version.
6. Indicate whether to export the common launch component. This component is required to run the application through Web Start, but a single instance of this can be deployed to the Web server and shared by multiple applications.
7. Optionally, you can have the export wizard sign the common launch component. The common launch component must be signed, but you can opt to sign it after it is exported using a utility such as the Java jarsigner.
8. You can select **Finish** to export the application using the default JNLP options, or select **Next** to bring up the JNLP options page, followed by the **Properties** page.

The default JNLP URL settings used by the export wizard will reference the application as exported to the destination directory. If you use these defaults, you can run your application directly from that directory by browsing to the main jnlp file in the directory. On Windows, you can also double-click on the jnlp file to run the application. You must have Web Start properly installed on your development machine for this to work. See “[Installing Web Start support on the client.”](#)

See “[Structure of a Client Services Web Start application” on page 174 for further information on the resources that are exported to the target directory. See “[Deploying the application on the server” on page 178 for information on how to deploy these resources on a web server.

**JNLP options**

The following table lists the JNLP options that can be specified during export. A default value of `<blank>` also implies that the option is not written to the JNLP file.

*Table 27. JNLP options*

<table>
<thead>
<tr>
<th>JNLP option</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application codebase</td>
<td>URL of location containing the application jar</td>
<td>Export destination</td>
</tr>
<tr>
<td>Common launch component URL</td>
<td>URL of the common launch component location</td>
<td>Location of the common launch component in the export destination directory</td>
</tr>
<tr>
<td>Title</td>
<td>Application title</td>
<td>Product configuration name</td>
</tr>
<tr>
<td>Vendor</td>
<td>Application vendor</td>
<td><code>&lt;blank&gt;</code></td>
</tr>
<tr>
<td>Description</td>
<td>Application description</td>
<td><code>&lt;blank&gt;</code></td>
</tr>
<tr>
<td>Offline allowed</td>
<td>If true, Web Start is allowed to run a cached version of the application if the client is not connected to the network</td>
<td>true</td>
</tr>
<tr>
<td>Homepage</td>
<td>URL to a homepage containing application information</td>
<td><code>&lt;blank&gt;</code></td>
</tr>
<tr>
<td>Icon</td>
<td>URL to an application icon</td>
<td><code>&lt;blank&gt;</code></td>
</tr>
</tbody>
</table>

**Properties**

The Web Start Export wizard **Properties** page allows you to specify name and value pair properties to pass to the application. These properties are passed as Java System properties, and can be accessed by the application through the System.getProperty and System.getProperties Java APIs.

**Installing Web Start support on the client**

This section describes how to install Web Start support on the client.

Installing Web Start support for Client Services applications on the client consists of the following three steps:

1. Configure the Lotus Expeditor Client for Web Start support.
2. Configure the browser for Web Start support.
3. Configure the client for Web Start support.

**Note:** It is important to apply these steps whenever the Lotus Expeditor Client is re-installed or updated.
Configuring the Lotus Expeditor Client

To properly configure Lotus Expeditor Client for Web Start support, perform the following steps:
1. Install the Lotus Expeditor Client.
2. Install J2SE 1.5 support for Expeditor through the DRE. J2SE 1.5 is required for running applications through Web Start.
3. Copy the rcpbootcp.jar to the J2SE 1.5 lib/ext directory.
   a. The rcpbootcp.jar can be found in the com.ibm.rcp.base plug-in at: `<Expeditor_Install>/rcp/eclipse/plugins/com.ibm.rcp.base_6.1.2.0`
   b. Copy rcpbootcp.jar to the following directory:
      On Windows: `<Expeditor_Install>/rcp/eclipse/plugins/com.ibm.rcp.j2se.win32.x86_1.5.0.<version>/jre/lib/ext`
      On Linux: `<Expeditor_Install>/rcp/eclipse/plugins/com.ibm.rcp.j2se.linux.x86_1.5.0.<version>/jre/lib/ext`

Additional configuration for Linux platforms

Depending on the IBM Lotus Expeditor capabilities required by your Web Start applications, additional updates to the environment might be needed on Linux:

• To support the Expeditor managed browser, set `MOZILLA_FIVE_HOME` to the appropriate Mozilla browser directory in /usr/lib. See “Client prerequisites” on page 7 for information on locating this directory on your system. For example:
  `export MOZILLA_FIVE_HOME=/usr/lib/firefox-1.5.0.3`

• If applications will be using DB2e, add the following location to the LD_LIBRARY_PATH environment variable:
  `<Expeditor>/rcp/eclipse/plugins/com.ibm.db2e.linux.x86_1.5.0.<version>/os/linux/x86`

• If applications will be using ISync, add the following location to the LD_LIBRARY_PATH environment variable:
  `<Expeditor>/rcp/eclipse/plugins/com.ibm.mobileservices.isync.linux.x86_1.5.0.<version>/os/linux/x86`

In the above paths, `<Expeditor>` is the install location of the Expeditor runtime and `<version>` is the version of the plug-in.

Configuring the browser

To configure the browser, you must install the Java 1.5 JRE on the client using the Java install program. The installation program for the JRE contains the logic to update the system and associate Java and Web Start with the browsers.

Once Java 1.5 has been installed, configure Web Start to reference the Expeditor platforms J2SE 1.5 as follows:

1. Open the Java Control Panel. On Windows, this can be found in the Windows Control Panel. On Linux, run application ControlPanel, found in jre/bin of the Java home directory.
2. On the Java tab, click View for the Java Application Runtime Settings.
   The JNLP Runtime Settings page is displayed.
3. On the User tab, click the Find button, and browse to the Expeditor J2SE 1.5 JRE. This will be at the following location:
   On Windows: `/rcp/eclipse/plugins/com.ibm.rcp.j2se.win32.x86_1.5.0.<version>`
On Linux:

/rcp/eclipse/plugins/com.ibm.rcp.j2se.linux.x86_1.5.0.<version>

4. Continue clicking Next until finished.

5. Uncheck all J2SE 1.5 JREs listed on the JNLP Runtime Settings page except for the Expeditor
   platform’s J2SE.

6. Click OK.

**Configuring the client**

To configure the Client Services Web Start support, create a configuration file that specifies where the
IBM Lotus Expeditor platform is located on the client. This configuration file allows multiple IBM Lotus
Expeditor-based platforms to be supported on a client machine. The configuration is an xml file with root
element <webstart> and one or more <platform> elements. Each <platform> element specifies the
location and optional version of a IBM Lotus Expeditor platform on the client. The following is an
example of a simple configuration specifying a single IBM Lotus Expeditor platform:

```xml
<webstart>
  <platform
    type="expeditor"
    version="6.1.1"
    location="C:/IBM/Lotus/Expeditor"/>
</webstart>
```

Set the environment variable RCP_JNLP_LAUNCH_CONFIG to the location of this configuration file.

**Deploying the application on the server**

This section describes the steps involved in deploying a Client Services Web Start application on the Web
server.

Perform the following steps to deploy a Client Services Web Start application on a web server:

1. Ensure that the Web server maps .jnlp files to mime type application/x-java-jnlp-file
2. Sign the common launch jar com.ibm.rcp.jnlp.launch_1.0.0.jar
3. Add the common launch jar as a resource to the Web server. This can either be in a common location
   referenced by all applications, or it can be copied into each application’s codebase directory. This
decision affects the contents of the application’s JNLP file. In particular, the resource must have its
href attribute properly set to the Common Launch component’s location.
4. Install the application jar file and JNLP files produced by the Web Start export to the Web server
   under the codebase specified in their JNLP files (or update the JNLP file’s codebase attribute to
   correspond to where you are placing them).

**Running an application through Web Start**

This section describes how to run an application through Web Start.

To run an application through Web Start, use a Web Start-enabled browser to browse to the jnlp file of
the application to be run.

**Internationalization considerations**

Considerations for using the characters that are not part of the ASCII character set.

- A Web Start application can be deployed locally on a client machine’s file system, rather than on a Web
  server. When this is done, the application’s codebase must only contain ASCII characters or characters
  that are part of the client machine’s current language.
- The Client Services Web Start export wizard cannot export to a file location containing characters that
  are not ASCII or not a part of the development machine’s current language.
The Client Services Web Start export wizard will automatically replace non-ASCII characters in an http: URL with their corresponding %xx URL encoding representation. This applies to the codebase and common launch component URL fields.
Managing the client runtime

The following sections provide general task information necessary to manage the client runtimes. Use the tasks specified in these sections as general patterns for managing the client runtimes.

You will need to refer to “Configuring the platform” on page 43 and “Extending the platform” on page 115 for the details to be supplied when completing each task.

Managing using the Device Manager server

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

The Enterprise Management Agent enables you to connect to a Lotus Expeditor Server. The Lotus Expeditor Server contains the Device Manager server component and provides basic platform management of the client, along with additional features, such as ISync and MQe servers.

Understanding managing with the Device Manager server

The Device Manager server provides capabilities for managing devices, including query and update functions.

The Enterprise Management Agent provided as part of the client platform is considered a device to the Device Manager server. Multiple installs on the same system will result in the appearance of multiple devices, each with individual account login information and device definition information.

To make use of the Enterprise Management Agent and the Device Manager server, follow these general steps:

1. Configure the Enterprise Management Agent on the device.
   - This consists of defining the address of the server, and the connection information, as well as polling parameters.
   - When the agent first contacts the server, the device will be enrolled, or registered to the server.
   
   The agent will periodically poll the Device Manager server to receive any jobs that need to be processed for the client.

2. Define jobs to the server to submit to one or more clients.
   - Jobs are tasks that need to be performed at the client.
   - Jobs can be assigned to one or more clients, including defined clients as well as enrolling clients.

Refer to the [Device Manager 2.1 information](#) topic in the Lotus Expeditor information center.

Configuring the Enterprise Management Agent

The Enterprise Management Agent is provided with Lotus Expeditor Client and enables you to manage the platform remotely using a Device Manager server.

The Enterprise Management Agent uses a Sample Account to connect to the server. There are four 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 9.
- After install time. Using the Enterprise Management Agent Servlet is described in the “Using the Enterprise Management Agent servlet” on page 187 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 Lotus Expeditor Client workbench. See “Configuring the agent using the desktop Enterprise Management Agent Preferences page.”

After install time. Using the OSGi Agent properties file to add or modify account and polling information.

Configuring the agent using the desktop Enterprise Management Agent Preferences page

This section describes how to configure the agent using the Enterprise Management Agent preference page. You can access the Enterprise Management Agent Preferences page using the File → Preferences → Enterprise Management Agent dialog. The following parameters appear on the Enterprise Management Agent Preferences page:

Table 28. Enterprise Management Agent Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Poll on Interval for Enterprise Management Agent</td>
<td>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.</td>
</tr>
<tr>
<td>Device User Name</td>
<td>The identity that a user’s machine uses to connect to a Device Manager server.</td>
</tr>
<tr>
<td>Device User Password</td>
<td>The password that a user’s machine uses to connect to a Device Manager server</td>
</tr>
<tr>
<td>Server Password</td>
<td>This is optional and depends on how the Device Manager server does authentication. It is only needed if the server requires it.</td>
</tr>
<tr>
<td>HTTP/HTTPS</td>
<td>Choose the type of connection with the server.</td>
</tr>
</tbody>
</table>
| 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:

     

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

     |
| 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 Manager server custom command job. |
Table 28. Enterprise Management Agent Parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling Start Time</td>
<td>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 Manager server custom command job.</td>
</tr>
<tr>
<td>Polling End Time</td>
<td>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 Manager server custom command job.</td>
</tr>
<tr>
<td>Enable Poll at Start Up</td>
<td>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 Manager server custom command job.</td>
</tr>
<tr>
<td>Test Connection</td>
<td>Select this button to test that the Enterprise Management Agent can connect to the Device Manager server using the parameters as specified on this page.</td>
</tr>
</tbody>
</table>

Note: The Lotus Expeditor Client preference panel does not allow creation of multiple accounts. The Device Manager 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 Lotus Expediter → Configuration Details.

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

Configuring the agent using the device Enterprise Management Agent Preferences page

This section describes how to configure the agent using the Enterprise Management Agent preference page on devices. You can access the Enterprise Management Agent Preferences page launching Application Manager and selecting Command → Preferences. The following parameters appear on the Enterprise Management Agent Preferences page:

Table 29. Enterprise Management Agent Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>The identity that a user’s machine uses to connect to a Device Manager server.</td>
</tr>
<tr>
<td>User Password</td>
<td>The password that a user’s machine uses to connect to a Device Manager server</td>
</tr>
<tr>
<td>HTTP/HTTPS</td>
<td>Choose the type of connection with the server.</td>
</tr>
</tbody>
</table>
Table 29. Enterprise Management Agent Parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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/OMADMServletAuthRequired`  
  If SyncML/DM authorization is not required by the server, append the following value to the Server Address instead:  
  `dmserver/OMADMServlet` |

**Note:** The Lotus Expeditor Client for Devices only supports one account. Any changes made in the preferences page or from the Device Management server to account information is applied to this one account.

### Configuring using the OSGi Agent properties file

The OSGi Agent is configurable using a properties file. The following are the supported keys and default values the agent uses at startup. Once the file is read the agent will remove the file. Under certain circumstances the property file is not deleted. If the agent is not being started for the first time, then the property file will only be read if the properties are specified for a new account. If no account name is specified in the properties file, the file is not read or deleted. To force the agent to read this property file again, the OSGiAgentTree.bin and its backup file must be deleted.

Before the agent starts for the first time, you can create a default account. When you first start the agent, it looks for the default account properties file, called osgiagent.properties, in the directory specified with a `-D` command line parameter in your invocation of the Virtual Machine (VM) that implements Java specifications. For example, if the account properties file is in the `c:\propertyfile` directory, you would use the following `-D` parameter in your VM invocation:

```
-Dcom.ibm.osgi.service.osgiagent.PropertiesFileLoc=c:\propertyfile
```

If the `-D` parameter is not used, the Java class path is searched for the osgiagent.properties file and that properties file is used. If no properties file is located in the Java class path, the current directory is searched.

When a properties file is found, the agent reads the parameters to create the initial account. When the OSGi agent starts for the first time, if no initial account exists, an initial account named SampleAccount is automatically created.

Use the following parameter descriptions and the sample file format as a guide if you create an osgiagent.properties file to create an initial account. The corresponding Agent Control Panel field is enclosed in parenthesis.

**AccountID (SyncML DM account ID)**

The SyncML DM account ID, which is often the same as the server ID. If no account name is specified, then SampleAccount is created the first time. However, if SampleAccount is already created, then the properties file will not be read if no AccountID is specified.
KeyRing (Key ring file name)
The file name for the key ring file.

The key ring file must be placed on the class path of the Windows 32-bit device in order for the
device agent to use SSL connections to the server.

Use a fully qualified path, including the file name. If the path is not specified, the current
working directory is used.

A key ring file stores the public keys of colleagues and associates in a single place for easy access
and retrieval.

UserName (Device user name)
User name for the device.
The name is used when SyncML/DM authorization or HTTP basic authentication is
implemented.

ClientPW (Device user password)
User password. The password text is displayed in clear text.
The password is used when SyncML/DM authorization is implemented.

KeyRingPassword (Key ring password)
The password for the key ring. The password is displayed in clear text.

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

Note: Security must be enabled in WebSphere Application Server.

Addr (Server address)
Use the host name or IP address for the Device Manager server, such as:
http://server_address/dmserver/OMADMServletAuthRequired

When authorization is required, use the following value:
dmserver/OMADMServletAuthRequired

If SyncML/DM authorization is not implemented, use the following value:
OMADMServlet

PollingEnabled (Polling enabled)
Determines if polling is enabled. The values are true or false (case is ignored).
The Polling enabled check box is displayed in the Device Information window.

PollingInterval (Polling interval)
Specifies the polling interval in hours and minutes. The value is specified as HH:MM.
This field is displayed in the Device Information window.

PollingStart (Polling start time)
Specifies the polling start time in hours and minutes. The value is specified as HH:MM.
This field is displayed in the Device Information window.

PollingEnd (Polling end time)
Specifies the polling end time in hours and minutes. The value is specified as HH:MM.
This field is displayed in the Device Information window.

LogSize (log size)
Specifies the size of the message files. This field is displayed in the Device Information window.
LogThreshold (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
   This field is displayed in the Device Information window.

TempFileLoc (Temp file location)
   Identifies the path for temporary files. This field is displayed in the Device Information window.

Configuring using the OSGi Agent properties file on the Lotus Expeditor Client for Devices

The OSGi Agent is configurable using a properties file. The following are the supported keys and default values the agent uses at startup. Once the file is read the agent will remove the file. Under certain circumstances the property file is not deleted. If the agent is not being started for the first time, then the property file will only be read if the properties are specified for a new account. If no account name is specified in the properties file, the file is not read or deleted. To force the agent to read this property file again, the OSGiAgentTree.bin and its backup file must be deleted.

Before the agent starts for the first time, you can create a default account. When you first start the agent, it looks for the default account properties file, called osgiagent.properties, in the directory specified with a -D command line parameter in your invocation of the Virtual Machine (VM) that implements Java specifications. For example, if the account properties file is in the \propertyfile directory, you would use the following -D parameter in your VM invocation (JVM_ARG):
-Dcom.ibm.pvc.osgiagent.core.propertiesfileloc=\propertyfile

found in the \eclipse\configuration\config.ini file.

If the -D parameter is not used, the Java class path is searched for the osgiagent.properties file and that properties file is used. If no properties file is located in the Java class path, the current directory is searched.

When a properties file is found, the agent reads the parameters to create the initial account. When the OSGi agent starts for the first time, if no initial account exists, an initial account named SampleAccount is automatically created.

Use the following parameter descriptions and the sample file format as a guide if you create an osgiagent.properties file to create an initial account.

Use the following fields to configure the Enterprise Management Agent properties on the first launch of the Lotus Expeditor Client for Devices:

Note: Table 30 lists the supported parameters. If unsupported parameters are used, you might receive exceptions when the osgiagent.properties is read by the client.

Table 30. Lotus Expeditor Client for Devices configurable properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Additional information</th>
</tr>
</thead>
</table>
Table 30. Lotus Expeditor Client for Devices configurable properties (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountID</td>
<td>SampleAccount</td>
<td>SampleAccount must be used or this property should not be included in the file. SampleAccount will be used by default if not included in the file.</td>
</tr>
<tr>
<td>UserName</td>
<td>User name for the device</td>
<td></td>
</tr>
<tr>
<td>ClientPW</td>
<td>User password</td>
<td></td>
</tr>
<tr>
<td>LogSize</td>
<td>Specifies the size of message files</td>
<td></td>
</tr>
<tr>
<td>LogThreshold</td>
<td>Specifies the minimum error type added to the message file</td>
<td>1 for Error 2 for Warning 3 for Info 4 for Debug</td>
</tr>
<tr>
<td>PollingEnabled</td>
<td>Determines if polling is enabled</td>
<td>True or False</td>
</tr>
<tr>
<td>PollingInterval</td>
<td>Specifies the polling interval in hours and minutes</td>
<td>Time in HH:MM format. For example, 00:05</td>
</tr>
</tbody>
</table>

Note: Lotus Expeditor Client for Device uses non-encrypted storage for this information.

Using the Enterprise Management Agent servlet
The Enterprise Management Agent servlet provides a user interface to update settings of the Enterprise Management Agent as well as perform administrative tasks.

Note: The Enterprise Management Agent servlet is not supported on Windows Mobile or Nokia devices. The Enterprise Management Agent servlet is not typically used by the end user and is provided as a tool to assist deployers in using the agent and the Device Manager server.

You can launch it from within the Lotus Expeditor Client workbench or from within a Web browser.

Installing the Enterprise Management Agent servlet:

The Enterprise Management Agent Servlet feature is provided on the main platform update site, but is not installed as part of the default platform.

If you want to use the Enterprise Management Agent servlet, you must install it before you can use it.

Starting the Enterprise Management Agent Servlet:

Follow these steps to start the Enterprise Management Agent Servlet:

To start the Enterprise Management Agent Servlet from within the Lotus Expeditor Client workbench, select Launcher > Enterprise Management Agent Servlet.

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 Manager 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

**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:

dmservlet/OMADMServletAuthRequired

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

OMADMServlet

Modifying Enterprise Management Agent account information using the properties file:

Device Manager supports a real-time notification of the OSGi device agents using the OSGi HTTP POST notification handler. The OSGi HTTP POST notification handler tracks the current IP address of each OSGi device agent.

This notification handler includes the following notification address attributes:
- IP address or host name (optional)
- Port number (optional); the default port is 8777
- Whether or not to have the client use the default account; the default is to use server ID instead

If you select to not use the default account, the server ID and server password associated with the device entry will be encoded and passed to the device as part of the notification.

Automatically update the notification IP address of the device; valid values: Yes or No. The default value is yes.

If the user decides not to automatically update the notification IP address of the device (No value), the IP address is required (specified above), and notification is sent to that IP address.

By default, Device Manager automatically fills-in the notification IP address for any newly enrolled OSGi devices. In addition, if the IP address has changed, the address is automatically updated each time the device connects to the Device Manager server. The device (client) listener port is assumed to be 8777, and is not updated. You must manage the port number if an alternative port is used.

To configure a Linux device or a Windows 32-bit device to use the OSGi HTTP POST notification handler, create the OSGiAgentServlet.properties file using a text editor. You may create it in any directory. A suggested location is the same directory as the launcher, which is:

<installation_directory> /rcp

The OSGiAgentServlet.properties file must meet all the requirements of a Java property file. Use the following parameters and values as a guide:
• AllowAddress provides a way to remotely access the user interface provided by OSGiAgentServlet. The AllowAddress parameter is not limited to the notification handler. The AllowAddress parameter only accepts an IP address.
• AllowExternalNotifications to allow external notifications, set the value to yes.
• AllowDefaultAccountForNotifications to allow the default account to receive notifications, set the value to yes.

Examples:
• AllowAddress=9.42.30.48
• AllowExternalNotifications=yes
• AllowDefaultAccountForNotifications=yes

Connecting to the server to check for jobs:

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

To connect to the Device Manager 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 Manager server to send a list of available software to the device.

Upon receiving a request from the device, the Device Manager 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 190.
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 Device Manager 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/OMADMServletAuthRequired

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

Default account
Defines the account that the agent uses to communicate with the Device Manager 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.

Changing the Enterprise Management Agent properties

A device configuration job can change the values for agent configuration parameters. You can change parameters for an individual device, several devices in the same device class, or all devices of a device class.
Refer to "Device Configuration Jobs" in the Device Manager server information center for additional details.

When a device enrolls with Device Manager, the device configuration parameters for that device can be set to initial values with a device configuration job. As appropriate, the device configuration parameters can be changed at a later date with another device configuration job. Refer to the Device Manager information center and Help for more information about device configuration and using the Device Manager console.

To make a device configuration job using the Device Manager console, do the following steps:
1. From the Device Manager console, select the target device, a group of devices, the device class, or criteria for running the job such as newly enrolled devices.
2. Right-click a device or the device class, and select Submit Job from the pop-up menu.
3. Use the Attributes window to specify the attributes for this job. The job attributes include the job type, activation date, expiration date, priority, and description, for example.
   Select Device Configuration in the Job Type field.
4. Use the Device Configuration Job Parameters window to set the values for the configuration parameters for this job.
   Use the tabs or the tree on the Parameters window to navigate to the parameters you want to set for this job.
   To view device parameter descriptions and their valid values for each supplied device class:
   a. Click the device class name in the Table of Contents.
   b. Click Configuration parameters.
5. Use the Submit Job Summary window to review and verify the target devices, attributes, and parameter information you specified for this job. Click OK to submit the job.

If the job is submitted successfully, the job ID for the created job is displayed. You can use the job ID to do other tasks such as:
   • View job history
   • View upcoming jobs
   • Exclude the device from jobs

**Registering features with the Device Manager server**
To deploy a feature using the Device Manager server, you will first need to define the feature and its contents to the server. Features must be defined within the context of an update site.

You can register a single feature from the update site or the entire update site. Refer to "Eclipse feature control job" in the Device Manager server information center for more information.

**Deploying Eclipse features**
Once the features have been registered to the Device Manager server, you can use the Device Manager server to create jobs to distribute the features to clients.

Refer to "Eclipse feature distribution job" in the Device Manager server information center for more information.

**Updating features with the Device Manager server**
When deploying new versions of applications, you will need to deploy new versions of features.

There are two models for updating deployed features on a target platform:
• If the feature descriptor (feature.xml) in the deployed feature contains an Update Site URL, then you can use the Eclipse Feature Control job to have the target platform attempt to update the feature from the feature update site. Refer to “Eclipse feature control job” in the Device Manager server information center. For more information, see “Registering features with the Device Manager server” on page 194.

• If the feature descriptor (feature.xml) in the deployed feature does not contain an Update Site URL, or you choose to deploy an updated version of the feature without referring to the update site referenced by the Update Site URL in the feature, then follow the install path for deploying the updated feature. You will first need to register the new feature, then install it. For more information, see “Deploying Eclipse features” on page 194.

Uninstalling Eclipse features
If you need to remove an application from the client, you can use the Device Manager server to perform this task. Refer to “Eclipse feature removal job” in the Device Manager server information center.

Deploying native applications
This section describes how to deploy native applications.

The command-line utility that creates native software bundles is called NativeAppBundle.bat. It is located on the Lotus Expeditor Client CD under utilis\NativeAppBundle. The parameter names and values for the utility can either be typed following the command or specified in a text file.

If a text file is used, specify the text file name using the following format:

NativeAppBundle.bat -Parameters=file_name

where file_name is the name of the text file containing the list of parameters. In the text file, there is one parameter and value per line. For more information, see “Parameters in a text file sample (NativeAppBundle command)” on page 200 and “Native software distribution jobs with OSGi bundles” when -Parameters is used, no other parameter names and values are specified when the NativeAppBundle.bat command is typed.

If the parameter names and values are typed after the command, use the following format:

NativeAppBundle.bat -InputDirectory=dir_name -BundleName=bundle_name
-InstallDirectory=dir_name -and so on

Required and optional parameters (NativeAppBundle command)
This section describes the required and optional parameters.

NativeAppBundle command parameters are separated into required and optional parameters.

Notes:

• When specifying path names, use “/” or “\” to specify directories.
• The NativeAppBundle command expects valid input. The command performs no input validation and results may be unexpected if invalid input is entered.
• When specifying a parameter value, such as yes or no, the value must be in the local language.
• Use the -Help parameter to view applicable values.

Required parameters (NativeAppBundle command)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-BundleName</td>
<td>Name of the native software bundle. The bundle name is required.</td>
</tr>
</tbody>
</table>
Table 31. Required NativeAppBundle parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-InputDirectory</td>
<td>Full path name of the directory for the files in the native software bundle. The input directory is required.</td>
</tr>
<tr>
<td>-InstallDirectory</td>
<td>Full path name of the directory on the device where the files in the native software bundle are installed. The install directory is required.</td>
</tr>
</tbody>
</table>

Optional parameters (NativeAppBundle command)

Table 32. Optional NativeAppBundle parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Use a # symbol at the first position of the line in the parameter file for writing comments.</td>
</tr>
<tr>
<td>-BuildDirectory</td>
<td>The name of the directory where the SMF JAR file is located.</td>
</tr>
<tr>
<td>-BundleVersion</td>
<td>The version number of the native software bundle. The default value is 1.0.0.</td>
</tr>
<tr>
<td>-BundleDescription</td>
<td>The description of the native software bundle. The default value is BundleName.</td>
</tr>
<tr>
<td>-JarName</td>
<td>The name of the native software bundle JAR file. The default value is BundleName.</td>
</tr>
<tr>
<td>-Prereq</td>
<td>Identifies the bundle name of the native software bundle which is a prerequisite to the bundle being installed.</td>
</tr>
<tr>
<td>-Install</td>
<td>Install program to run on the device after all the files in the native software bundle have been installed. The full path name of the program on the device must be specified.</td>
</tr>
</tbody>
</table>

The -Install option is only invoked if the bundle is distributed with the Auto Start parameter set to True.

Parameters can be passed to the program.

Note: The Lotus Expeditor Client for Devices does not support the -Install parameter.

-PreInstall and -PostInstall

Specify a program to run before, after, or both, the installation program (specified by -Install) runs.

You can use one or more of the following options:

- -PreInstall=file_name_to_run
- -PostInstall=file_name_to_run

Optionally, installation verification steps can be included with the -PostInstall parameter.

-WhenToRunPostInstall

Determines when the post-install program is run. The allowed values are:

- always  Always run the post-install program
- afterInstallFailure  Only run the post-install program if the installation fails.
- afterInstallSuccess  Only run the post-install program if the installation is successful.
### Optional NativeAppBundle parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-Uninstall</strong></td>
<td>Uninstall program to run on the device prior to the installed files being removed when the native software control bundle is uninstalled. The full path name of the program on the device must be specified. Parameters can be passed to the program. <strong>Note:</strong> The Lotus Expeditor Client for Devices does not support the -Uninstall parameter.</td>
</tr>
<tr>
<td><strong>-RemoveOnUninstall</strong></td>
<td>Determines whether files are removed when the native software control bundle is uninstalled. The files to be removed are in the directory identified by the -InstallDirectory parameter. The valid values are yes or no. The default value is yes.</td>
</tr>
<tr>
<td><strong>-CleanupAfterInstall</strong></td>
<td>Determines whether files are removed when the standard native software bundle installation program is installed. The files to be removed are in the directory identified by the -InstallDirectory parameter. The valid values are yes or no. The default value is no.</td>
</tr>
<tr>
<td><strong>-Start</strong></td>
<td>Start program to run on the device every time the native software control bundle is started. When a native software bundle is distributed, the bundle state is at an “installed” state when Auto Start is set to True. Use a bundle control job with a Start action to start the native software control bundle. The full path name of the program on the device must be specified. Parameters can be passed to the program. The <strong>-Start</strong> option is not invoked during the bundle start of the initial distribution of the package. If the <strong>-Start</strong> option has been started by using the bundle control job and is not stopped before the OSGi run-time is stopped, every restart of OSGi run-time will restart -Start option.</td>
</tr>
<tr>
<td><strong>-Stop</strong></td>
<td>Stop program to run on the device every time the native software control bundle is stopped. The full path name of the program on the device must be specified. Parameters can be passed to the program. Use a bundle control job with a “stop” action to stop the native application control bundle.</td>
</tr>
</tbody>
</table>
Table 32. Optional NativeAppBundle parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-IgnoreNonZeroInstallRC</td>
<td>Use the return code failure parameter (-IgnoreNonZeroInstallRC) to determine whether or not to fail the job if the return code for the install step (INSTALL_EXIT_VALUE) is non-zero. Some commands use non-zero return codes to indicate a successful condition. The allowed values are: yes: Fail the job (default value) no: Do not fail the job If the install step fails because of a non-zero return code (INSTALL_EXIT_VALUE), the device is restarted only if a restart was specified after the corresponding install step. For pre-install (PRE_INSTALL_EXIT_VALUE) and post-install (POST_INSTALL_EXIT_VALUE) steps, a non-zero return code always indicates a job failure. When a job failure is detected, further steps are not attempted (except if INSTALL_EXIT_VALUE is non-zero and -IgnoreNonZeroInstallRC=no).</td>
</tr>
<tr>
<td>-WorkingDirectory</td>
<td>The path to the working directory. The default is current directory. You must have R/W/X access to the working directory. If the specified working directory does not exist, a new directory is created and deleted after use. All files created should be deleted after use.</td>
</tr>
<tr>
<td>-PrereqVersion</td>
<td>The minimum version number of the prerequisite bundle. The format for the version must be: n n.n n.n.n If the prerequisite version is not specified, it defaults to the most recent version. The automatic prerequisite resolution tool chooses at least the specified version number. For example, if the specified version is 1.7.1 and the available versions are 1.7.0, 1.7.1 and 1.7.2, by default, 1.7.2 would be download.</td>
</tr>
<tr>
<td>-Help</td>
<td>Displays the program syntax. All other parameters are ignored.</td>
</tr>
</tbody>
</table>

Related concepts

["Job options and tasks for distributing native software bundles (OSGi bundles)"]

This section describes the job options and tasks for distributing native software bundles.

["Native software distribution jobs with OSGi bundles” on page 201]

This section describes native software distribution jobs with OSGi bundles.

Job options and tasks for distributing native software bundles (OSGi bundles)

This section describes the job options and tasks for distributing native software bundles.

To distribute a native software bundle, use the following job options and do the following tasks:

- Identify the native software for the native software bundle.
- Optionally, identify the installation program to run on the device after all the files in the native software bundle have been installed (-Install parameter).
- Optionally, identify the uninstall program to run on the device prior to the installed files being removed when the native software control bundle is uninstalled (-Uninstall parameter).
- Optionally, identify the start program to run on the device every time the native software control bundle is started (-Start parameter).
Optionally, identify the stop program to run on the device every time the native software control bundle is stopped (-Stop parameter).

Optionally, include the capture and storage of return codes, stdout, and stderr messages. The allowed values are YES (capture and store messages) and NO (default value).

If CAPTURE_OUTPUT=YES, the return codes, stdout, and stderr messages are captured and then stored as details in the job completion record for the device. The captured output has the following format:

```
PRE_INSTALL_EXIT_VALUE: return_code
PRE_INSTALL_STDOUT: stdout
PRE_INSTALL_STDERR stderr

INSTALL_EXIT_VALUE: return_code
INSTALL_STDOUT: stdout
INSTALL_STDERR stderr

POST_INSTALL_EXIT_VALUE: return_code
POST_INSTALL_STDOUT: stdout
POST_INSTALL_STDERR stderr
```

Optionally, use the return code failure parameter (-IgnoreNonZeroInstallRC) to determine whether or not to fail the job if the return code for the install step (INSTALL_EXIT_VALUE) is non-zero. Some commands use non-zero return codes to indicate a successful condition.

Create the native software bundle using NativeAppBundle.bat. Review all of the parameters for NativeAppBundle.bat, and include the parameters applicable for your native software distribution job.

Put the native software bundle JAR file on the HTTP server.

When registering the software using the Device Manager console, point to the newly created native software bundle. Ensure the software location on the HTTP server is accessible to the Device Manager server using HTTP or HTTPS protocol.

If you are using the Device Manager console to create jobs, use the Native software bundle job type when you distribute the OSGi bundles to the target devices.

### Running the installation more than once

If the standard native software bundle installation program is installed again while the native software control bundle is installed:

- The specified files are not be downloaded to the device again.
- The standard native software bundle installation program is not run.

Optionally, you can remove the native software control bundle from the device with an “uninstall” program. Use the -Uninstall parameter of NativeAppBundle.bat to specify this uninstall program. After the native software control bundle is uninstalled, the standard native software bundle installation program can be installed again and the native software bundle with the native software can be installed again.

### Related information

- Native software distribution with OSGi bundles
- Required and optional parameters
- Sample for parameters in a text file
- Software distribution with OSGi bundles
- Controlling bundles on a device
Parameters in a text file sample (NativeAppBundle command)
When distributing native software bundles, you can specify the parameters in a text file or on the command line with the NativeAppBundle command.

Sample for specifying the parameters in a text file

The command to create the native software bundle from a text file is:
```
NativeAppBundle.bat -Parameters=parameterList
```

The parameterList text file has the following lines:
```
-BundleName=TestBundle2
-InputDirectory=c:/SourceFiles/Bundle2
-InstallDirectory=c:/Program Files/testBundles
-BundleVersion=2.1.0
-LocalStorageInBytes_c:/=100000000
-LocalStorageInBytes_n:/=50000000000
-Install=c:/Program Files/testBundles/InstallPgm.exe -parm1
-Uninstall=c:/Program Files/testBundles/UninstallPgm.exe
-Start=c:/Program Files/testBundles/bundleProcess.exe
-RemoveOnUninstall=yes
-JarName=DemoTestBundle
-WhenToRunPostInstall=always
```

The NativeAppBundle.bat -Parameters=parameterList command and parameter text file does the following actions:
- Creates a native software bundle named DemoTestBundle+2_1_0.jar (-BundleVersion and -JarName parameters).
- The JAR file contains all files in the c:/SourceFiles/Bundle2 directory (-InputDirectory parameter).
- The bundle is named TestBundle2 (-BundleName parameter).
- After the standard native software bundle installation program is installed and started on the device, the files are installed with InstallPgm.exe and it passes "parm1" to the installation (-Install parameter).
- The standard native software bundle installation program installs the files into the c:/Program Files/testBundles directory on the device (-InstallDirectory parameter).
- The amount of local storage needed is at least 100 megabytes of free space on the c:/ path and at least 50 megabytes free on the d:/ path (-LocalStorageInBytes parameter).
- The uninstall program is c:/Program Files/testBundles/UninstallPgm.exe. It runs on the device prior to the installed files being removed when the native software control bundle is uninstalled (-Uninstall parameter).
- Every time the native software control bundle is started, the c:/Program Files/testBundles/ bundleProcess.exe start program runs (-Start parameter).
- The installed files are removed when the native software control bundle is uninstalled (-RemoveOnUninstall parameter).

Sample for typing parameters on a command line

The native software bundle command-line utility can be typed on a single line (for readability, the command below appears on several lines):
```
NativeAppBundle.bat -BundleName=NativeTestBundle
               -InputDirectory=c:/SourceFiles/Bundle1
               -InstallDirectory=c:/Program Files/testBundles
```

The NativeAppBundle.bat command does the following actions:
- The JAR file contains all files in the c:/SourceFiles/Bundle1 directory (-InputDirectory parameter).
- The bundle is named NativeTestBundle (-BundleName parameter).
• The native software control bundle has the default name (NativeTestBundle). The native software control bundle is installed, but not started.
• The standard native software bundle installation program installs the files into the c:/Program Files/testBundles directory on the device (-InstallDirectory parameter).
• The installed files are removed when the native software control bundle is uninstalled. The default value for RemoveOnUninstall parameter is yes.
• There is no install program, uninstall program, or start program.

Native software distribution jobs with OSGi bundles
This section describes native software distribution jobs with OSGi bundles.

For OSGi devices, such as Windows 32-bit devices, Linux devices, and the devices that Lotus Expeditor Client supports, software can be distributed as OSGi bundles. An OSGi bundle is comprised of Java classes and other resources which together can provide functions to device owners and provide services and packages to other bundles. A bundle is distributed as a JAR file.

To distribute native software with OSGi bundles, the Lotus Expeditor Client includes a command-line utility to wrap the native software in an OSGi bundle for distribution. This bundle is called a native software bundle. The native software bundle is placed on a Web server and registered with the Lotus Expeditor Server’s Device Manager to make the bundle available for software distribution.

The device agent gets the OSGi bundle directly from the URL associated with the registered software. The location is specified by the URL associated with the registered software.

The native software in the bundle is installed into the desired directory on the device using the standard native software bundle installation program. An optional installation program can be specified and run on the device. After the optional installation program runs, the native software bundle automatically stops and uninstalls itself.

A separate native software bundle used for native software bundle control is automatically installed on the device before the native software bundle removes itself from the device. The device agent uses this native software control bundle to do a software inventory of the native software bundle. The native software control bundle is named BundleName (same name as the native software bundle).

When this native software control bundle is started, an optional “start” program can be started (and stopped) numerous times. Each time the native software control bundle starts, the optional start program is run.

Native software distribution jobs with OSGi bundles are available for the following OSGi device classes:
• Linux devices
• Windows 32-bit devices
• Windows Mobile 2003 Standard Edition or Windows Mobile 5 devices

For native software distribution jobs with OSGi bundles, the bundle name, bundle version, and bundle description from the manifest file are used for the software package name, version, and description in the Device Manager database.

A native software distribution job does not automatically update the software inventory in the Device Manager database for the target devices. To update that database, run an inventory collection job for the target devices. For more information, see “Device Manager 2.1” in Using Lotus Expeditor Server.

Native software distribution intended for Device Manager agents use the Native software distribution job type.
Job progress limitation for native applications

When a native software distribution jobs is run, the job progress indicates success, failure, or other job progress conditions. However, the job progress does not provide the job progress condition of a native application running on the device.

For example, if as part of the distribution of a native application, you use the -Install parameter to run an installation program, the success or failure of that installation program cannot be monitored with the job progress information. Similarly, if a bundle control job is used to start an installation program, the success or failure of that installation program cannot be monitored with the job progress information. The job progress reports only on the status of the Device Manager job. For more information, see “Device Manager 2.1” in Using Lotus Expditer Server.

Related information
- Software distribution jobs with OSGi bundles
- Controlling bundles on a device

Managing preferences

Extensions to the SyncML/DM management tree are provided by the client platform that enable management of the Eclipse preferences.

The preferred mode for working with Eclipse preferences is through the use of the Eclipse preferences jobs.

An alternative to using the Eclipse preferences jobs is to use the Node discovery job and Custom command jobs.

Querying client runtime preferences

You can query the runtime preference values to assist in configuring the Lotus Expditer Client.

To query the current preferences, use the Eclipse preferences retrieval job. Refer to “Eclipse preferences retrieval job” in the Device Manager server information center.

You can also use the Node Discovery job to query the client runtime preferences. Use ./Configuration/Preferences/Eclipse as the root of the Target URI. Preferences information retrieved through the Node Discovery job will be viewable in the Management Tree tab of the View Inventory in the console. Refer to “Node discovery jobs” in the Device Manager server information center.

Creating client runtime preferences

Use the Device Manager server Eclipse preferences editing job to create new preferences.

Refer to “Eclipse preferences editing job” in the Device Manager server information center for more information.

You can also use Custom command jobs to create preferences:
- Begin any Target URI with ./Configuration/Preferences/Eclipse/ before adding your specific preference node and key information.
- When creating nodes, use text/plain for the Meta Type and node for the Meta Format
- When creating leafs (keys), use text/plain for the Meta Type and chr for the Meta Form.

Refer to “Custom command jobs” in the Device Manager server information center for more information.
Updating client runtime preferences
Use the Device Manager Server Eclipse preferences editing job to update preferences.

Refer to “Eclipse preferences editing job” in the Device Manager server information center.

You can also use Custom command jobs to update preferences. Begin any Target URL with .//Configuration/Preferences/Eclipse/ before adding your specific preference node and key information. Refer to “Custom command jobs” in the Device Manager server information center.

Deleting client runtime preferences
Use the Device Manager server Eclipse preferences editing job to delete preferences.

Refer to “Eclipse preferences editing job” in the Device Manager server information center.

You can also use Custom command jobs to delete preferences. Begin any Target URL with .//Configuration/Preferences/Eclipse/ before adding your specific preference node and key information. Refer to “Custom command jobs” in the Device Manager server information center.

Managing properties files
Extensions to the Enterprise Management Agent enable the updates of individual property values within Java properties files.

Files that can be managed must by registered using extension points. See Extending the platform for Device Manager properties management on page 134.

The Device Manager Console provides specific jobs for the management of properties. These are the preferred jobs.

You can also use custom command jobs to work with properties.

Querying property file values
Use the Properties retrieval job to query the property file values on the client.

Refer to “Properties retrieval job” in the Device Manager server information center.

You can also use a Node Discovery job to retrieve properties information. Use .//Configuration/Properties as the beginning of the Target URL. Results from a Node discovery job are displayed in the Management Tree tab when selecting View Inventory on a device. Refer to "Node discovery jobs" in the Device Manager server information center.

Changing a property file value
Use the Properties editing job to change a value of an existing property in a property file.

Refer to “Properties editing job” in the Device Manager server information center.

You can also use a Custom command job to perform a property value update. Be sure to include .//Configuration/Properties before the property file name and property name. Refer to "Custom command jobs" in the Device Manager server information center.

Adding a new property to a property file
Use the Device Manager server Properties editing job to create new properties in a defined properties file.

Refer to “Properties editing job” in the Device Manager server information center for more information.

You can also use Custom command jobs to create a new property in a defined properties file.
• Begin any Target URL with ./Configuration/Properties/ before adding your specific property file name and property key information.
• When creating keys, use text/plain for the Meta Type and chr for the Meta Form.

Refer to “Custom command jobs” in the Device Manager server information center.

Deleting a property from a property file

Use the Device Manager server Properties editing job to delete existing properties in a defined properties file.

Refer to “Properties editing job” in the Device Manager server information center.

You can also use Custom command jobs to delete an existing property in a defined properties file. Begin any Target URL with ./Configuration/Properties/ before adding your specific property file name and property key information. Refer to “Custom command jobs” in the Device Manager server information center.

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 (for example, 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® Alliance 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 Manager server using the Device Manager console application.

1. Click Devices.
2. Select Use New Query and Return anything as your search criteria, and then click OK. The Device Manager 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.

Configuring the Web Container using the Enterprise Management Agent:

You can remotely change the configuration of the Web Container using the Enterprise Management Agent.

Note: The following are prerequisites to the configuration procedure:

- ConfigurationAdmin store has already been pre-populated with Web container configuration data.
- The Enterprise Management Agent is registered with the Device Manager server and is ready to go.

Follow these steps 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 Manager server using the Device Manager console application.

1. Click Devices.

2. Select Use New Query and Return anything as your search criteria, and then click OK. The Device Manager 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. Once the job has completed, select the device, then click View Inventory.

10. Then select Management Tree.

11. Select the configuration entry you want to configure and click Submit Job.

12. Click Next. Leave the defaults as supplied in the panel, then click Next.

13. Enter 1 in 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 Rich Clients using the Portal server

You can enable your Lotus Expeditor Client workstations to receive configuration information from a WebSphere Portal server. This allows you to centrally administer your Lotus Expeditor Client environment.
Centrally administered role-based access control is a critical requirement of all large enterprises. WebSphere Portal provides a set of powerful functions specifically targeted at these requirements. The Managed Browser Administration Portlet exposes each instance of the browser as a portlet. Additionally, configurable elements of the browser instance (such as address bar, URL restrictions, for instance) are also exposed through the portlet. By exposing these configurable elements, they can now be managed in a role-based fashion. Integrating with WebSphere Portal allows the overall user administration to remain as unified as possible, which reduces both financial and personnel cost.

Administrators can use the same page and place management, user management, and access controls to integrate, manage, and reuse application components running across a variety of device types, ranging from pervasive devices such as cell-phones and PDAs to browser-based desktops. The Portal-administered client extends this reach further to support “rich client” desktop devices based on Lotus Expeditor Client.

The term Portal-administered client simply means a side of the Lotus Expeditor Client that can install, load, and run applications defined by Portal. Portal applications are a specific kind of application that ultimately means a collection of pages, portlets, policies and roles. A WebSphere Portal application by definition is much different than other applications, for example, a rich client application in terms of Eclipse. The equivalent of a Portal application on the rich client is a set of navigable perspectives (pages) that contain views (portlets) and are configured with Eclipse preferences (policy) and can communicate through a property broker (wiring).

A Portal portlet maps to an Eclipse view, not a client portlet. This view can be any proper Eclipse view resident on the client. One of the views included with the client is a viewer to render HTML output from a real portlet running in the portlet container on the client. This portlet viewer view is treated in special ways on the client and in the administration portlets so it is easy to get a little loose with terminology and say a portlet maps to a portlet.

Portal gives you the ability to define a series of pages for your application and a navigation hierarchy for how those pages are represented. What this means is every single application can actually define a complete hierarchy of pages all contained within a single Portal application. This further implies that a projected Portal application on the rich client has \( n \) number of perspectives associated with that application.

```
Root Page 1
Root Page 2
  Sub Page 2.1
  Sub Page 2.2
Label 3
  Label-Link 3.1
  Sub Page 3.2
    Sub Sub Page 3.2.1
    Label-Link 3.2.2
```

WebSphere Portal applications running on the client need extended client side properties that need to be specified on the pages and portlets on the server. To make the process of specifying these attributes easier, use the following portlets. These portlets would expose Properties and Actions that map to the Eclipse counterpart on the client. The WAR file would then need to have an equivalent SWT plug-in that uses the same WSDL for the client-side property broker registration. This usage is basically SWT applications that are aggregated on Portal and deployed from Portal.

**Rich Client Layout Administration Portlet**

This is an extension to the Portal page customizer that is used to manage common rich client-specific metadata. It can inject this data into any portlet on any page as portlet preferences. A typical application with one or more pages may include instances of several types of portlets. The placeholder and administration proxies serve to establish position of portlets on pages that project as views and perspectives on the client. JSR168 portlets also project as views but can in addition have client-side equivalents that run in the portlet container. Since JSR168 portlets do not have to be explicitly instrumented to manage rich client specific data, the same rich client
unaware portlet can serve as an administration proxy for versions deployed at the rich client and be used as a "normal" portlet on the server for delivery through a browser. The Rich Client Layout Administration Portlet will not manage view-specific attributes such as wiring properties/actions or view configuration. If this is needed for a given view, an appropriately programmed administration proxy will be needed.

The Generic Placeholder portlet

This portlet is used to aggregate any generic eclipse view that does not expose properties and actions, or require view specific custom configuration. This portlet is little more than a container for the common layout preferences managed by the Rich Client Layout Administration Portlet (for example, view ID, ratio, feature requirements). This portlet is included in the web module with the Rich Client Layout Administration Portlet and can be copied and renamed as necessary by the administrator, then placed on pages to represent views.

"Administration proxy" portlet

Unlike the generic placeholder portlet, an administration proxy is tightly coupled to the rich client view it represents. It may expose "dummy" properties and actions so that the portlet wiring tool can be used to create wires between administration proxies that are then sent in the composite application XML from the Portal server. This assumes the plug-in developer has implemented an action that can scope the wire to a specific view/secondary id within the plug-in. This portlet may also provide a custom JSP that is used to manage view-specific configuration properties. Lotus Expeditor Client includes two administration proxies; the Managed Browser Administration portlet and the Rich Client WSRP Viewer Enablement Portlet. In general an administration proxy must be created by the developer as part of the process of developing client-side plug-ins that support wiring and/or view configuration.

These portlets are installed on the Portal server by the Network Client Installer. See "Installing with the Network Client Installer" on page 29 for information about the Network Client Installer.

This section describes how to manage the Lotus Expeditor Client using the WebSphere Portal server.

Related tasks

"Configuring Managed Settings" on page 70
The managed settings framework allows you control over the runtime behavior of client applications by letting you set the values of the settings that the applications are reading out of the Eclipse preference store.

Understanding managing using the Portal server

The Lotus Expeditor Client runs under Eclipse so the display window is laid out mostly using perspective containers with the content generated by views.

The Rich Client Layout Administration portlet supports server administration of view parameters and view placement within a perspective. At a high level this maps nicely to portlets placed on a page. The hierarchical arrangement of pages on Portal has no direct equivalent on Eclipse but the effect is achieved through the Lotus Expeditor Client code that understands both arrangements.

At a detailed level the Eclipse environment is more complex so several parameters must be added to the Portal configuration to effectively represent the client layout. For example, under Eclipse there can be functions called activities started automatically when a perspective is opened. Portal pages have no similar concept so the client code recognizes an additional page parameter to specify the activities. The Portal server aggregates the output of portlets on a page automatically and a web browser renders the page according the HTML generated by the server and space available in the window on the client. Under Eclipse the views can be controlled more precisely by specifying relative ratios of the total perspective space, so the client code recognizes an additional portlet parameter for the ratio.

The Rich Client Layout Administration portlet supplied with the Lotus Expeditor Client provides a GUI to configure the common parameters for pages and perspectives and portlets and views. Other portlets
address Eclipse preferences, custom view parameters for Managed Browsers, a WSRP client enabler and the Lotus Expeditor Client contributions that are not configured as views under perspectives.

The layout administration on Portal consists of arranging pages and portlets hierarchically under a single Content Root. You can add pages starting directly under the root or to one of the pages predefined when the Portal is installed. The Lotus Expeditor Client retrieves information only from under Content Root → Application Root. This is where the Portal server places its Composite Applications. However, the administrator will normally use the Template Library functions under the Templates link on the Home page. Here you can import and create application templates, then use them to create Composite Applications in the Application Library. Applications can also be divided into categories with the Lotus Expeditor Client normally requesting the list of applications in the MyClient category.

Regardless of the starting point, each page can then have pages under it and portlets on it. During normal operation, child page selection is displayed according to a theme with tabs being common. The content generated by the portlets fills the remainder of the window. Note that there is also a label element, basically a page that has child pages but no portlets, and a URL element that is a link to outside the Portal or sideways to an existing page. The root of a composite application is technically a label, but since this document concerns the layout and content of pages with portlets, the label and URL are mentioned if appropriate but not discussed further.

In Portal, pages are artifacts that exist only within the content of items in a database. The associated page properties and container layout are also information kept by the server in a database. The page content references real portlets that execute code to produce the rendered output seen in a Web browser. Likewise, Eclipse has metadata to describe perspectives and their contained views, and the content of the views comes from code in plug-ins written as SWT applications. The original IBM API portlet execution environment and the base SWT environment are very different so the content generating code is also very different. This is changing as a portlet container in the Lotus Expeditor Client can run newer standard portlets (JSR 168 compliant) such that portlets not using any Portal specific functions or Eclipse specific function can run in either. But the Eclipse environment remains more flexible and complex so additional page and layout information still must be added to what Portal keeps to accurately describe layout of Eclipse elements.

**Using the IBM Lotus Expeditor MyClient sample template**

As part of the Network Client Installer, you can use the MyClient template that is added to your Portal server to help you to deploy composite applications.

When you create a new application using this template, the template automatically creates a Workbench Contribution page. Use this page to place administration proxy portlets with parameters defining dynamic extensions on the client. The Rich Client Workbench Custom Eclipse Administration and Rich Client Workbench Native Application Administration portlets are examples of portlets that generate dynamic extensions. This page is marked as hidden by default, so if you want to contribute to the Lotus Expeditor Client user interface using this template, you should create another page and place the content on that page.

**Configuring client user interface layout**

Portal has a set of page properties that it uses itself and has a GUI to edit them.

The server allows additional properties for a page as needed for client administration and has a generic page parameter editing GUI, but it requires you to know the exact spelling of each parameter name and manually ensure constraints on the values are met.

Since the parameters are processed remotely on a client, errors can be difficult and time consuming to diagnose. Part of the Rich Client Layout Administration portlet is to provide a GUI to edit the new page parameters added for use in the Portal Administered Client. This GUI provides descriptive titles and ensure constraints are met.
Handling of the content producing elements is more complicated. For layout purposes on Portal, there must be a portlet instance to represent each view instance under Eclipse. This surrogate portlet will have parameters for the additional layout information and be projected as an Eclipse view on the client. There is a standard set of parameters that the Rich Client Layout Administration portlet will create and edit for any portlet. Beyond those the surrogate portlet must be an administration proxy with its own configuration capability instead of just a placeholder relying entirely on the Rich Client Layout Administration portlet.

**Using the Rich Client Layout Administration Portlet**

The Rich Client Layout Administration Portlet is an extension to the Portal page customizer that is used to manage common rich client-specific metadata.

It can inject this data into any portlet on any page as portlet preferences. A typical application with one or more pages may include instances of several types of portlets. The placeholder and administration proxies serve to establish position of portlets on pages that project as views and perspectives on the client. JSR168 portlets also project as views but can in addition have client-side equivalents that run in the portlet container. Since JSR168 portlets do not have to be explicitly instrumented to manage rich client specific data, the same rich client unaware portlet can serve as an administration proxy for versions deployed at the rich client and be used as a "normal" portlet on the server for delivery through a browser. The Rich Client Layout Administration Portlet will not manage view-specific attributes such as wiring properties/actions or view configuration. If this is needed for a given view, an appropriately programmed administration proxy will be needed.

The Rich Client Layout Administration Portlet does not manage view-specific attributes such as wiring properties and actions or view configuration. If this is needed for a given view, you must use an administration proxy.

The Rich Client Layout Administration Portlet handles the parameters that are common to all perspectives and views:

**Page Properties**

These are defined once on each page and stored in the Portal parameters for the page.

**Page portlets**

This is a list of portlets on the page and is generated when needed based on the current page layout configuration.

**Rich Client Properties**

These describe execution properties of an Eclipse view on the client. They are stored in the portlet settings and portlet data.

**Feature Requirements**

These are prerequisite features the client will need before starting the target view. They are stored in the portlet settings and portlet data.

**Layout Properties**

These are the additional parameters needed to account for the Eclipse layout being more granular and complex than Portal. They are stored in portlet settings and portlet data.

Before you begin, make sure the Rich Client Layout Administration Portlet is installed on the Portlet server. The portlets are normally installed by running an appropriate installer under the /nci directory on the product CD. The portlets are present in directories under /nci/portal and can be installed manually by using the Portal administration dialog to install war files from /nci/portal/portlets, or by running xmlaccess scripts under /nci/portal/scripts. Additionally, you must have previously created a page structure to represent an application intended for client configuration.

**Administering rich client-specific page parameters:**
You can edit page parameters that are applicable to rich clients. Once you have changed and stored the parameters, the client software can access and process the changes.

Follow these steps to administer page parameters that are specific to your enabled Portal-administered client environment.

1. Log in to WebSphere Portal as an administrator.
2. Click Administration in the Portal toolbar.
3. In the navigation tree, click Portal User Interface, then Manage Pages.
4. Navigate to the page layout of the target page.
5. Click the Rich Client tab. The page parameters that can be changed, along with their current values, are displayed.
6. Change the parameters in the Page Parameters section.
7. Click OK to save the changes.

Rich client specific page parameters are changed and stored, ready for the client software to access them.

**Page properties**

Page properties are automatically inherited from ancestor pages by the Portal Server and Topology Handler on the client. The Rich Client Layout Administration Portlet does not recognize or use the inheritance information. Any parameter you change plus certain required parameters are always written for the page on which they are entered. This is different from the Portal Server page parameter setting which indicates inheritance and allows any you have changed to revert to the inherited values.

The following provides detailed information on the fields found in this section of the page properties page. The Visible and Hidden radio buttons select between two mutually exclusive states for displaying the perspective represented by this page.

**Visible**

Indicates that the perspective is created and can be made accessible through the navigator and/or the launcher.

Access through the navigator on the client is controlled by a parameter that is not configured through this portlet. A parameter named com.ibm.rcp.useNavigator can be placed on the root label of the containing application. If this parameter is present and the value set to “true”, all visible perspectives and views projected from pages and portlets will be available through the navigator on the client.

To enable the navigator view, use Portal Server Administration → Portal User Interface → Manage Pages → Content Root → Application Root → Edit Page Properties (page icon) for target application → Advanced options → I want to set parameters. Add the new parameter com.ibm.rcp.useNavigator with a value of true.

- **Add to the launcher** – If checked, the perspective represented by this page is added to the launcher in the client user interface.
- **Open automatically when the application starts** – The projected perspective opens with no action from the user required. This is only applicable if Add to the launcher is also selected.

**Hidden**

Indicates that the perspective is not shown on either the navigator or the launcher. This should be selected when the page has admin proxy portlets with parameters defining dynamic extensions on the client. The Rich Client Workbench Custom Eclipse Administration and Rich Client Workbench Native Application Administration portlets are examples of portlets that generate dynamic extensions.

- **Icon for application switcher** - Specifies the path to an application branding icon that is specified relative to the base WebSphere Portal application. This is used by the application switcher to represent the perspective and cached locally on the client.
- **Activities** – Specifies which activities to include in the perspective when it is started on the client. The values are the unique identifiers of Eclipse activity extensions. When there is more than one value separate the values with a semicolon.

- **6.1.2 Page Alias** – Specifies a key identifier that enables you to access the page programmatically in the Expeditor runtime.

- **Default folder for views not specified in the layout** – Defines the default folder in which to display any views which do not have an associated folder ID specified in portlet properties.

The following table lists the page parameters:

<table>
<thead>
<tr>
<th>Page parameter</th>
<th>GUI line label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.rcp.activities</td>
<td>Activities</td>
<td></td>
</tr>
<tr>
<td>com.ibm.rcp.defaultFolder</td>
<td>Default folder for views not specified in the layout</td>
<td>The same as the Icon configurable with the regular admin interface in Portal 6.0.</td>
</tr>
<tr>
<td>com.ibm.portal.PageIcon</td>
<td>Icon for application switcher</td>
<td></td>
</tr>
<tr>
<td>com.ibm.rcp.launcher</td>
<td>Add to the launcher</td>
<td>Boolean</td>
</tr>
<tr>
<td>com.ibm.rcp.autoStart</td>
<td>Open automatically when the application starts</td>
<td>Boolean</td>
</tr>
<tr>
<td>com.ibm.rcp.hidden</td>
<td>Hidden</td>
<td>Boolean</td>
</tr>
<tr>
<td>com.ibm.rcp.useNavigator</td>
<td>Hidden</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

**Creating a copy of the Rich Client View Placeholder before adding it to a page:**

You can create a copy of the Rich Client View Placeholder and use it for any view not needing custom parameters. The placeholder is also used for theportlet viewer.

To create a copy of a Rich Client View Placeholder in a perspective:

1. Make sure the Rich Client Layout Administration portlet is properly installed on WebSphere Portal.
2. Log on to WebSphere Portal using administrator's credentials.
3. Click **Administration** under **Product Links**.
4. Expand **Portlet Management** and click **Portlets**.
5. Locate the portlet named Rich Client View Placeholder.

   **Tip:** To quickly find the Rich Client View Placeholder, select the default search **Starts with** in the **Search by** field and type *r* before clicking the **Search** field.

6. Click the copy portlet icon to create a copy.

**Related concepts**

"Adding the Rich Client View Placeholder to a page"

You can add the Rich Client View Placeholder to a page so that the selected page has an additional portlet to represent a rich client view.

**Adding the Rich Client View Placeholder to a page:**

You can add the Rich Client View Placeholder to a page so that the selected page has an additional portlet to represent a rich client view.

Follow these steps to add the Rich Client View Placeholder to a page:
1. Log in to WebSphere Portal as an administrator.
2. Click **Administration** in the Portal toolbar.
3. In the navigation tree, click **Portal User Interface**, then **Manage Pages**.
4. Navigate through the pages or search for the target page until it appears in the list of pages.
5. Click the Edit Page Layout icon (the pencil) on the line listing the target page.
6. Click **+Add portlets**.
7. Scroll through or search until the desired portlet shows in the list.
8. Enable the box before the portlet title.
9. Click **OK** to save the changes.
10. Click **Done** to exit editing page layout.

The Rich Client View Placeholder now appears on the page layout.

**Editing common rich client properties for a portlet on a page:**

You can administer rich client properties for a portlet on a page. The set of properties to be edited is the same for all portlets.

Follow these steps to edit rich client properties.

1. Log in to WebSphere Portal as an administrator.
2. Click **Administration** in the Portal toolbar.
3. In the navigation tree, click **Portal User Interface**, then **Manage Pages**.
4. Navigate through the pages or search for the target page until it appears in the list of pages.
5. Click the Edit Page Layout icon (the pencil) on the line listing the target page.
6. Click the **Rich Client** tab.
7. Scroll through or search until the desired portlet shows in the list. This section provides a list of the portlets on the current page. The **Portlet Title** shown is for the portlet (that is, the concrete portlet) and is the same for all placements of that portlet on all pages. Each portlet can have a unique description called the **Portlet Instance Description**.
8. Enable the box before the portlet title.
9. Clicking a Configure icon (wrench) takes you to a page to modify the properties associated with the selected portlet and that are applicable to a client view.
   Clicking an Edit icon (pencil) takes you to a similar page to edit portlet properties but the changes are stored as portlet instance data for this page instead of portlet settings for the portlet. These icons do not put the Rich Client Layout Administration Portlet into the respective modes; they represent the logical operation of the Rich Client Layout Administration Portlet relative to the data for the target portlet.
10. Click **OK** to save the changes.
11. Click **Done** to exit editing page layout.

Rich client parameters for the target portlet are changed and stored, ready for the client software to access them.

The following table list the view attributes you can edit.

**Note:** This portlet page also adds the Instance description field provided in the previous list of portlets presented in the portlet's first page

The properties in this section represent characteristics of an Eclipse view. The values are not arbitrary but must reflect the actual characteristics of the corresponding entity on the client.
Instance description

This is an optional field that can be used to distinguish between multiple instances of the same portlet on one page. It is metadata used for administration using the Rich Client Layout Administration Portlet and does not display on the client.

Extension type

Indicates how the Topology Handler on the client will process the portlet’s parameters. This field is not editable but its value controls how Rich Client Layout Administration Portlet processes certain form fields. The only recognized values are:

Simple view
The portlet represents a view described by only its Eclipse view ID. This is the correct value for a portlet that represents an SWT (Standard Widget Toolkit) view on the client.

Local portlet
The portlet represents a standard (JSR 168 compliant) portlet running locally in the portlet container on the client. The view configured here is actually an instance of the portlet viewer on the client that connects to the local portlet through a URL (context root).

WSRP portlet
The portlet represents an instance of the portlet viewer on the client and that viewer connects to a remote portlet on a server. The connection is made using parameters set by the WSRP admin proxy portlet.

Dynamic extension
The portlet represents a dynamic extension where the XML stream defining the extension is supplied by an admin proxy portlet rather than being constructed by the Topology Handler. The Rich Client Workbench Custom Eclipse Extension and Rich Client Workbench Native Application portlets are examples of this.

The extension type is used by the Rich Client Layout Administration Portlet to control the presentation of the SWT view ID and local portlet context root as described later. This lets you see an accurate representation of how the parameters are processed by Topology Handler on the client.

The radio buttons let you select between two mutually exclusive types of view. For each type, the applicable set of properties is listed below the selection line.

This portlet represents an SWT view on the rich client

Represents an SWT view which is the basic or lowest level interface normally used by an Eclipse application.

- Eclipse view ID - The unique identifier of the Eclipse view extension represented by this portlet.
- Variations are based on the value of extension type. When Rich Client Layout Administration Portlet detects extension type is normal (lower case), normal editing is allowed. When the extension type has been set by another portlet to read only (upper case), no change is allowed.

The following table illustrates the variations:

<table>
<thead>
<tr>
<th>Extension Type</th>
<th>View Selection Radio Button</th>
<th>View ID in user interface</th>
<th>View ID Type Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMPLE VIEW</td>
<td>Not shown</td>
<td>Display only</td>
<td>No change</td>
</tr>
<tr>
<td>simple view</td>
<td>Selected</td>
<td>Editable</td>
<td>Add or change</td>
</tr>
<tr>
<td>LOCAL PORTLET</td>
<td>Not shown</td>
<td>Not shown</td>
<td>Delete if present</td>
</tr>
<tr>
<td>local portlet</td>
<td>Selectable</td>
<td>Not shown</td>
<td>Delete if present</td>
</tr>
<tr>
<td>wsrp portlet</td>
<td>Not shown</td>
<td>Not shown</td>
<td>Delete if present</td>
</tr>
<tr>
<td>dynamic extension</td>
<td>Not shown</td>
<td>Not shown</td>
<td>Delete if present</td>
</tr>
</tbody>
</table>
Table 34. Extension types (continued)

<table>
<thead>
<tr>
<th>Extension Type</th>
<th>View Selection Radio Button</th>
<th>View ID in user interface</th>
<th>View ID Parameter Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present but invalid</td>
<td>Not shown</td>
<td>Display</td>
<td>No change</td>
</tr>
<tr>
<td>Not present</td>
<td>Selectable</td>
<td>Editable</td>
<td>Change or add</td>
</tr>
</tbody>
</table>

This portlet runs locally on the rich client

Represents a JSR 168 compliant portlet running in the portlet container included in the Expeditor client. If you are using only standard portlet APIs, this portlet can be exactly the same as runs on a Portal server. Whether or not the portlet is the same, it is developed for the server then packaged in a bundle for the client using the Expeditor WAB tool.

- Portlet context root - Additional information required for the client portlet viewer to connect to the portlet running in the client portlet container. A portlet running on the Portal Server does not have a context root because on the server a URL points to a page and all portlets on the page are aggregated by the server.
- Variations are based on the value of extension type. When RCLAP detects the extension type is normal (lower case), normal editing is allowed. If another portlet has set the extension type to read only (upper case), no change is allowed. The following table illustrates the variations:

Table 35. Extension types

<table>
<thead>
<tr>
<th>Extension Type</th>
<th>Portlet Selection Radio Button</th>
<th>Context Root in user interface</th>
<th>Content Root Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMPLE VIEW</td>
<td>Not shown</td>
<td>Not shown</td>
<td>No change</td>
</tr>
<tr>
<td>simple view</td>
<td>Selectable</td>
<td>Not shown</td>
<td>No change</td>
</tr>
<tr>
<td>LOCAL PORTLET</td>
<td>Not shown</td>
<td>Read only</td>
<td>No change</td>
</tr>
<tr>
<td>local portlet</td>
<td>Selected</td>
<td>Editable</td>
<td>Change or add</td>
</tr>
<tr>
<td>wrsp portlet</td>
<td>Not shown</td>
<td>Not shown</td>
<td>No change</td>
</tr>
<tr>
<td>dynamic extension</td>
<td>Not shown</td>
<td>Not shown</td>
<td>No change</td>
</tr>
<tr>
<td>Present but invalid</td>
<td>Not shown</td>
<td>Display only</td>
<td>No change</td>
</tr>
<tr>
<td>Not present</td>
<td>Selectable</td>
<td>Editable</td>
<td>Change or add</td>
</tr>
</tbody>
</table>

Table 36. Portlet parameters

<table>
<thead>
<tr>
<th>Line label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal view</td>
<td>Specifies that the view is placed normally by the Topology Handler using the following parameters:</td>
</tr>
<tr>
<td>Folder ID</td>
<td>The unique ID of the eclipse tabbed folder to which this view should be added when it is added to the perspective layout.</td>
</tr>
<tr>
<td>Visible when perspective is shown</td>
<td>If selected, the view is visible when the application is first started on the client. If not, a placeholder is created for it. When the view is opened, the view appears in the placeholder position.</td>
</tr>
<tr>
<td>Show the view title bar</td>
<td>If selected, the resulting view is displayed normally with the title bar. If this is not selected, the view displays without the title bar.</td>
</tr>
<tr>
<td>Allow this view to be moved within the perspective</td>
<td>If selected, the view can be closed or moved.</td>
</tr>
<tr>
<td>Add this view as standalone</td>
<td>If selected, this view as standalone.</td>
</tr>
<tr>
<td>Fast view</td>
<td>If selected, the view is added to the layout as a fast view. Fast views are added to the fast view bar in the client.</td>
</tr>
</tbody>
</table>
Table 36. Portlet parameters (continued)

<table>
<thead>
<tr>
<th>Line label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio</td>
<td>The portion of the layout area this component should occupy based on the component it is attached to within the layout. This is the Eclipse based sizing of perspective layouts. If this is not specified the default ratio constant supplied by Eclipse is used. Ratio is a decimal number in the range of .05 to .95. For example, if you want to set up four portlets in a column and each with an equal height, how do you compute the ratio? Note that the ratio parameter you set in a portlet is not the real ratio of itself, but the ratio of its previously placed neighbor. For the first portlet, as it lies on the top of all portlets, it does not need a ratio. For the second one, it should have a ratio, but the ratio is set for the first one, which may take 1/4 space of the whole column, so the ratio is 0.25. For the third one, it should have a ratio for the second one, which may take 1/3 of the remainder space, so the ratio is 0.33. For the last one, it should have a ratio for the third one. As this status, there are only 2 portlets left, each may take half of the remainder space, so the ratio should be 0.5.</td>
</tr>
<tr>
<td>Allow this view to be closed from its title bar</td>
<td>If selected, the client user can close the view from the title bar. If not selected, the close function is not present on the title bar. If there is no title bar the close function is not available.</td>
</tr>
</tbody>
</table>

The following table lists the attributes for a view. Under PAA they must be stored in settings and/or instance data exactly as recognized by the client Topology Handler. Note that this table includes both the Rich Client Properties and Layout Properties as described above.

Table 37. PAA (Topology Handler) Portlet Data Parameters

<table>
<thead>
<tr>
<th>PAA (Topology Handler) Portlet Data Parameter</th>
<th>GUI Line Label</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.rcp.instanceDescription</td>
<td>Instance Description</td>
<td>Not used on clientk</td>
</tr>
<tr>
<td>com.ibm.rcp.extensionType</td>
<td>Extension type</td>
<td>Enumerated. Valid values are: • simple view • local portlet • wsrp portlet • dynamic extension</td>
</tr>
<tr>
<td>com.ibm.rcp.viewId</td>
<td>Eclipse view id</td>
<td>As entered, except see com.ibm.rcp.extensionType</td>
</tr>
<tr>
<td>com.ibm.rcp.viewId</td>
<td>portlet runs locally</td>
<td>Client uses com.ibm.rcp.portletviewer.portletview</td>
</tr>
<tr>
<td>com.ibm.rcp.portletcontainer.contextRoot</td>
<td>Portlet context root</td>
<td>Portlet viewer not Topology Handler</td>
</tr>
<tr>
<td>com.ibm.rcp.fastView</td>
<td>Fast view</td>
<td>Boolean</td>
</tr>
<tr>
<td>com.ibm.rcp.folder</td>
<td>Folder ID</td>
<td></td>
</tr>
</tbody>
</table>
Table 37. PAA (Topology Handler) Portlet Data Parameters (continued)

<table>
<thead>
<tr>
<th>PAA (Topology Handler) Portlet Data Parameter</th>
<th>GUI Line Label</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.rcp.visible</td>
<td>Visible when perspective is shown</td>
<td>Boolean</td>
</tr>
<tr>
<td>com.ibm.rcp.showTitle</td>
<td>Show the view title bar</td>
<td>Boolean</td>
</tr>
<tr>
<td>com.ibm.rcp.moveable</td>
<td>Allow this view to be moved within the perspective</td>
<td>Boolean</td>
</tr>
<tr>
<td>com.ibm.rcp.standalone</td>
<td>Add this view as stand-alone</td>
<td>Boolean</td>
</tr>
<tr>
<td>com.ibm.rcp.ratio</td>
<td>Ratio</td>
<td></td>
</tr>
<tr>
<td>com.ibm.rcp.closeable</td>
<td>Allow this view to be closed</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

Editing or adding a feature requirement for the portlet representing an Eclipse view:

Administer a feature requirement for the portlet representing an Eclipse view. This allows you to specify the Eclipse features that are needed as prerequisites for this portlet to render properly on the client.

Each portlet maps to an Eclipse-based view (SWT view) and the features required to enter that view must be loaded on the client first. The features you specify will be automatically loaded using the Eclipse Update Manager from the URL specified.

The steps assume you have placed portlets intended to represent views on the rich client.

1. Log in to WebSphere Portal as an administrator.
2. Click Administration in the Portal toolbar.
3. In the navigation tree, click Portal User Interface, then Manage Pages.
4. Navigate to the page layout of the target page.
5. Click the Rich Client tab. The page parameters and a list of portlets are displayed.
6. Click the configure (wrench) or edit (pencil) icons on the line of the portlet whose prerequisites are to be changed. The page for modifying rich client properties of the target portlet is displayed.
7. In the Feature Requirements section, click Add or the modify icon on the line for an existing feature ID. The page to edit properties for a feature requirement is displayed with all fields either empty or filled in for a specific requirement, depending on whether add or modify was selected.
8. Change the input fields as appropriate.
9. Click an action button to save the changes.
10. Click either OK or Done to finish and return to the portlet properties. Click Add More if present to create another feature requirement.

Rich client feature requirements parameters for the target portlet are changed and stored, ready for the client software to access them.
Note: If the version of a feature is modified to a new version in the Portal editor, the Composite Application may need to be restarted when it is first loaded from the client. This allows activation of the new version of the feature in the client to take place.

The following feature parameters are available when adding or modifying a feature:

**Feature ID**
Defines the Eclipse ID for the required prerequisite feature.

**Feature version**
Defines the desired version number of the required feature.

**Matching rule**
Defines the version filtering algorithm used by the Eclipse update manager. It installs or updates the specified feature ID only if it finds one whose version compares to the specified version according to the matching rule specified. The following values are in the selection list box:

- **perfect**
  The actual version must be an exact match to the specified version.

- **equivalent**
  The actual version must be the same as the version specified or at a higher service level (major and minor version levels must equal the specified version).

- **compatible**
  The actual version must be the same as the version specified or at a higher service level or higher minor level (major level must equal the specified version).

- **greaterOrEqual**
  The actual version must be the same as the version specified or at a higher service level or higher minor or higher major level.

**Provisioning URL**
Specifies the URL of the Eclipse Install/Update site where the client is to fetch this required feature. If not specified a default value will be supplied by the provisioning component on the client.

The following table shows the key formats for requirements, using the sample data above. The numeric suffix of the feature parameter names serves to associate the parameters for each feature and allow multiple features.

<table>
<thead>
<tr>
<th>PAA (Requirements Manager) Portlet Data Parameter</th>
<th>GUI Line Label</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>id.Feature_1</td>
<td>Feature id</td>
<td></td>
</tr>
<tr>
<td>version.Feature_1</td>
<td>Feature version</td>
<td></td>
</tr>
<tr>
<td>match.Feature_1</td>
<td>Matching rule</td>
<td></td>
</tr>
<tr>
<td>url.Feature_1</td>
<td>Provisioning URL</td>
<td></td>
</tr>
<tr>
<td>shared.Feature_1</td>
<td>Boolean where true means to put in shared location and false means to install as user specific.</td>
<td></td>
</tr>
<tr>
<td>remove.Feature_1</td>
<td>Boolean where true means to remove the feature.</td>
<td></td>
</tr>
</tbody>
</table>

Removing a feature requirement for the portlet representing an Eclipse view:

You can delete feature requirements for a rich client view in a perspective as represented by a portlet on a page.
The steps assume you have placed portlets intended to represent views on the rich client.
1. Log in to WebSphere Portal as an administrator.
2. Click **Administration** in the Portal toolbar.
3. In the navigation tree, click **Portal User Interface**, then **Manage Pages**.
4. Navigate to the page layout of the target page.
5. Click the **Rich Client** tab. The page parameters and a list of portlets are displayed.
6. Click the configure (wrench) or edit (pencil) icon on the line of the portlet whose prerequisites are to be changed. The page for editing rich client properties of the target portlet is displayed.
7. In the **Feature Requirements** section, click the delete icon (trash can) on the line for an existing feature ID. You will be prompted for confirmation of the delete action. Click **Yes** to delete the feature requirement. The changes are saved.

The rich client feature requirement for the target portlet is removed, ready for the client software to access new configuration. This does not cause the feature to be deleted from the client if it has already been installed. It merely removes it from the list for this view.

**Using the Managed Browser Administration portlet**

The Managed Browser Administration Portlet enables WebSphere Portal to centrally manage instances of the Managed Browser view on the client.

Each instance of the Managed Browser Administration Portlet represents an instance of the managed browser on the Lotus Expeditor Client. Multiple instances of the Managed Browser Administration Portlet can be placed on a single WebSphere Portal page to aggregate multiple distinct instances of the managed browser in an Eclipse perspective. The Managed Browser Administration Portlet allows WebSphere Portal administrators to access control the capabilities of the Managed Browser based on user role. When viewed from an HTML capable agent, the portlet provides an interface for configuring the client side browser instance.

You can perform the following tasks with the Managed Browser Administration Portlet:

- "Creating an instance of a Managed Browser Administration Portlet in a perspective"
- "Creating a client side wire between the managed browser and another wire-capable portlet" on page 221
- "Using the Managed Browser Administration Portlet to run a local Web application" on page 223

**Related concepts**

- "Installing the Managed Browser Administration Portlet"
  Install the Managed Browser Administration Portlet on a WebSphere Portal server.

**Installing the Managed Browser Administration Portlet:**

Install the Managed Browser Administration Portlet on a WebSphere Portal server.

Portlets are installed by the Lotus Expeditor Client server-based client installer. See "Installing with the Network Client Installer" on page 29 for information about installing a portlet.

**Creating an instance of a Managed Browser Administration Portlet in a perspective:**

You can create a new instance of the Managed Browser Administration Portlet and configure it to represent the browser instance that is aggregated in the perspective.

To create an instance of a managed browser in a perspective:
1. Make sure the Managed Browser Administration Portlet is properly installed on WebSphere Portal.
2. Log on to WebSphere Portal using administrator’s credentials.
3. Click Administration under Product Links.
5. Locate the portlet named Managed Browser.

   Tip: To quickly find the Managed Browser portlet, select Title contains in the Search by field and specify Managed Browser in the Search field.
6. Click the copy portlet icon to create a copy.
7. Name your copy of the portlet as appropriate and click OK.
8. Place the new copy on the appropriate perspective page.
9. Click the Portal down-arrow and select Configure from the drop down menu.
10. Configure the browser settings. See “Managed browser settings” for a list of the browser settings you can edit.
11. Click Submit to save the changes and return to the view mode.

**Managed browser settings**

You can edit the following Managed Browser settings.

*Table 39. Managed Browser general settings*

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>Controls which browser engine is used to render the Web pages.</td>
<td>MSIE – Microsoft Internet Explorer Mozilla – Mozzia browser Platform – Platform default</td>
</tr>
<tr>
<td>Initial URL</td>
<td>URL the browser should open at startup.</td>
<td>Any valid URL address</td>
</tr>
<tr>
<td>URL points to local web application</td>
<td>Defines whether the URL points to a local Web application.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Use Browser Home</td>
<td>Determines if the underlying browser’s home page should override the initial URL specified.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Popup Style</td>
<td>Determines the popup style for the new window.</td>
<td>Default/Popup/Embedded</td>
</tr>
</tbody>
</table>

*Table 40. Managed Browser user interface settings*

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show URL</td>
<td>Determines whether the address bar is be displayed.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Show Toolbar</td>
<td>Determines whether the browser toolbar is to be displayed.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Back and Forward Buttons</td>
<td>Determines whether the Back and Forward buttons are to be displayed.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Show Print Button</td>
<td>Determines whether the Print button is to be displayed.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Show Home</td>
<td>Determines whether the Home button is to be displayed.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Show Bookmarks</td>
<td>Determines whether the favorites/bookmark icon is to be displayed.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Show Page Control</td>
<td>Determines whether the Stop/Refresh page buttons are to be displayed.</td>
<td>Yes/No</td>
</tr>
</tbody>
</table>
**Table 41. Managed Browser security settings**

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow Script Execution</td>
<td>Determines if a Web page can execute JavaScript™</td>
<td>Yes/No</td>
</tr>
<tr>
<td>URL Domains List</td>
<td>A list of URL domains that are either a white list or a black list.</td>
<td>A list of URLs, entered one per line.</td>
</tr>
<tr>
<td>URL Domains Permission</td>
<td>Determines whether the URLs specified in the URL Domains List are a black list (when true) or a white list. If the domains list is defined as a white list, then the browser will only be able to navigate to the URLs listed. If the domains list is defined as a black list, then the browser can navigate to any URL not on the list.</td>
<td>Allow only these URLs/Block these URLs.</td>
</tr>
<tr>
<td>Popup Domains List</td>
<td>A list of URL domains that is either a white list or a black list that defines sites can bring up popup windows.</td>
<td>A list of URLs, entered one per line.</td>
</tr>
<tr>
<td>Popup Domains Permissions</td>
<td>Determines whether the domains list specified in the Popup Domains List is a black list (when true) or a white list. If the domains list is defined as a white list, then the browser will allow popups only from those sites listed. If the domains list is defined as a black list, then the browser allows popups from all sites except those listed.</td>
<td>Allow only these URLs/Block these URLs.</td>
</tr>
<tr>
<td>Enable Applets</td>
<td>Determines whether to support Java2 applets.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Enable Bookmarks</td>
<td>Determines whether Bookmark functions are enabled.</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Controlled Download Domains</td>
<td>A list of domains for restricting unsolicited file downloads.</td>
<td>List of URLs, entered one per line.</td>
</tr>
<tr>
<td>Controlled Download Path</td>
<td>A local path to save the download file in domains defined by Controlled Download Domains.</td>
<td>String</td>
</tr>
<tr>
<td>Allow Download Override</td>
<td>Determines whether a new download is allowed to overwrite an existing file with the same name on the file system.</td>
<td>Yes/No</td>
</tr>
</tbody>
</table>

**Table 42. Managed Browser proxy settings**

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy Autoconfig URL</td>
<td>Specifies the automatic proxy configuration URL. If this item is set, other proxy settings will be ignored.</td>
<td>String</td>
</tr>
</tbody>
</table>
Table 42. Managed Browser proxy settings (continued)

<table>
<thead>
<tr>
<th>Configuration Item</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy Server</td>
<td>Specifies the unified proxy server. If this item is set, proxies for separate network protocol will be ignored (HTTP Proxy Server, HTTP Proxy Port, FTP Proxy Server, FTP Proxy Port, and SSL Proxy Server).</td>
<td>String</td>
</tr>
<tr>
<td>Proxy Port</td>
<td>Specifies the unified proxy port. If Proxy Server is set, this setting along with setting the Proxy Server item will be used for all network protocols.</td>
<td>Integer</td>
</tr>
<tr>
<td>HTTP Proxy Server</td>
<td>Specifies the proxy server for HTTP protocol. This setting item works when Proxy Autoconfig URL and Proxy Server are not set.</td>
<td>String</td>
</tr>
<tr>
<td>HTTP Proxy Port</td>
<td>Specifies the proxy port for HTTP protocol.</td>
<td>Integer</td>
</tr>
<tr>
<td>FTP Proxy Server</td>
<td>Specifies the proxy server for FTP protocol. This setting item works when Proxy Autoconfig URL and Proxy Server are not set.</td>
<td>String</td>
</tr>
<tr>
<td>FTP Proxy Port</td>
<td>Specifies the proxy port for FTP protocol.</td>
<td>Integer</td>
</tr>
<tr>
<td>SSL Proxy Server</td>
<td>Specifies the proxy server used for SSL. This setting item only works when Proxy Autoconfig URL and Proxy Server are not set.</td>
<td>String</td>
</tr>
<tr>
<td>SSL Proxy Port</td>
<td>Specifies the proxy port for SSL protocol.</td>
<td>Integer</td>
</tr>
<tr>
<td>Socks Proxy Server</td>
<td>Specifies the socks proxy server address. This setting item only works when Proxy Autoconfig URL and Proxy Server are not set.</td>
<td>String</td>
</tr>
<tr>
<td>Socks Proxy Port</td>
<td>Specifies the proxy port for socks protocol.</td>
<td>Integer</td>
</tr>
<tr>
<td>Proxy Bypass domains</td>
<td>Specifies a list of domains that do not need to go through the proxy server.</td>
<td>List of URLs, entered one per line.</td>
</tr>
</tbody>
</table>

Creating a client side wire between the managed browser and another wire-capable portlet:

You can wire the Managed Browser Administration Portlet to another portlet.

Wires allow portlets to communicate. They are used to save property transfers between portlets.

The other portlet must either represent an Eclipse view on the client side or be a JSR 168 portlet that is meant to be run on the client-side JSR 168 container.

To wire the Managed Browser Administration Portlet to another portlet:
1. Make sure instances of the Managed Browser Administration Portlets are already created.
2. Create another portlet with a WSDL that defines the wiring properties that has been installed on the Portal server and properly registered with the property broker.
3. Log on to WebSphere Portal using administrator’s credentials.
4. Click Administration under Product Links.
5. Expand Portlet Management and click Portlets.
6. Locate the portlet named Managed Browser.

   Tip: To quickly find the Managed Browser portlet, select Title contains in the Search by field and specify Managed Browser in the Search field.

7. Click the copy portlet icon to create a copy.

8. Name your copy of the portlet as appropriate and click OK.

9. Place the new copy on the appropriate perspective page.

10. Place an instance of another wire-capable portlet on the same page.

11. Edit the page layout.

12. From the edit page layout, click the Wires tab.

13. Choose a source portlet.

14. Choose a property from the source portlet.

15. Choose a target portlet.


17. Click Done to complete the process.

Managed Browser output properties for 6.1.1

6.1.1 The Managed Browser portlet provides the following input and output properties – please note that these properties will be deprecated in future releases:

Table 43. Output properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DummyGetURLAction</td>
<td>Notifies user when browser URL is changed.</td>
</tr>
<tr>
<td>DummyGetTitleAction</td>
<td>Notifies user when the title is changed.</td>
</tr>
<tr>
<td>DummyOpenWindowAction</td>
<td>Notifies user when a new browser view is open.</td>
</tr>
<tr>
<td>DummyCloseWindowAction</td>
<td>Notifies user when the browser view is closed.</td>
</tr>
<tr>
<td>DummyDocumentModelAction</td>
<td>Notifies user when the Web page is loaded.</td>
</tr>
</tbody>
</table>

Table 44. Input properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setURLAction</td>
<td>Set URL to browser and load.</td>
</tr>
<tr>
<td>TextAction</td>
<td>Set HTML text for browser.</td>
</tr>
<tr>
<td>StopAction</td>
<td>Stops browser loading current page.</td>
</tr>
<tr>
<td>SetCookie</td>
<td>Sets cookie on browser.</td>
</tr>
</tbody>
</table>

Managed Browser output properties for 6.1.2

6.1.2 Lotus Expeditor versions 6.1.2 uses standardized data types. The Managed Browser portlet provides the following input and output properties:

Table 45. Output properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>currentURL</td>
<td>Notifies the user when a URL has changed.</td>
</tr>
<tr>
<td>changedTitle</td>
<td>Notifies the user when the title is changed.</td>
</tr>
</tbody>
</table>
### Table 45. Output properties (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewViewID</td>
<td>Notifies the user when a new browser view is open.</td>
</tr>
<tr>
<td>ClosedViewID</td>
<td>Notifies the user when the browser view is closed.</td>
</tr>
<tr>
<td>StatusText</td>
<td>Notifies the user when a new Web page is loaded.</td>
</tr>
<tr>
<td>DocumentLoadedStatus</td>
<td>Notifies the user when a page’s status has changed.</td>
</tr>
</tbody>
</table>

### Table 46. Input properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Set the target URL to load.</td>
</tr>
<tr>
<td>HTMLText</td>
<td>Set the target HTML text content to load.</td>
</tr>
<tr>
<td>StopMessage</td>
<td>Stop loading the current Web page.</td>
</tr>
<tr>
<td>CookieData</td>
<td>Set cookie data with the specified URL.</td>
</tr>
</tbody>
</table>

### Using the Managed Browser Administration Portlet to run a local Web application:

You can configure the Managed Browser Administration Portlet to load the Web application.

To run a local Web application:
1. Make sure instances of the Managed Browser Administration Portlets are already created.
2. Enter configure mode for the instance of Managed Browser Administration portlet.
3. Specify values for the following fields.
   - **Browser**
     Controls which browser engine is used to render the Web pages. Select from MSIE (Internet Explorer), Mozilla (Mozilla browser), or Platform (Platform default).
   - **Home page URL**
     Context root of the local web application. This URL should be in the form of http://localhost/mycontextRoot or https://localhost/mycontextroot. If the URL begins with the https:// protocol, it is assumed that SSL is used.
   - **Use browser home page**
     Set to No. This ensures the home page URL is used as the homepage of the browser.
   - **URL points to local web application**
     Set to Yes.
   - **Popup window style**
     Specify the Popup style for the new window.
4. Click **Submit** to save the changes and return to view mode.

### Using the Workbench Administration portlet

The Workbench Administration portlet enables WebSphere Portal to manage items contributed to the various portions of the Lotus Expeditor Client workbench. These portlets provide a way to make user role based contributions to the workbench.

The Workbench Administration portlet allow you to centrally manage and access control items contributed to the following four areas of the Lotus Expeditor Client workbench:

- Tool bar
- Status bar
- Application launcher
The Workbench Administration portlet supports contributing the following items:

- Native applications
- Password changer
- Locale switcher
- Keyboard layout switcher
- Shutdown
- Logoff
- Lockup
- Clock
- Task list

You can perform the following tasks with the Workbench Administration portlet:

- "Contributing an instance of a native application to the launcher"
- "Contributing the screen lock function to the workbench" on page 226
- "Contributing the change password function to the workbench" on page 226
- "Contributing the change locale function to the workbench" on page 227
- "Contributing the Tasklist function to the workbench" on page 228
- "Contributing the change keyboard function to the workbench" on page 228
- "Contributing the system shutdown function to the workbench" on page 229
- "Contributing the system logoff function to the workbench" on page 230
- "Contributing the status bar clock to the workbench" on page 230
- "Contributing a custom dynamic eclipse extension to the workbench" on page 231
- "Assigning access control to launcher items" on page 231

Related concepts

- "Installing the Workbench Administration portlet on a WebSphere Portal server"

Portlets are installed by the server-based client installer.

Installing the Workbench Administration portlet on a WebSphere Portal server:

Portlets are installed by the server-based client installer.

See "Installing with the Network Client Installer" on page 29 for information about installing a portlet.

Contributing an instance of a native application to the launcher:

You can create a new instance of the Native Application Administration portlet and configure it to represent the native application that appears on the launcher.

To contribute an instance of a native application to the launcher, follow these steps.

1. Make sure at least one Workbench Administration portlet is properly installed on WebSphere Portal.
2. Log on to WebSphere Portal using administrator's credentials.
3. Click Administration under Product Links.
5. Locate the portlet named Workbench Native Application Administration.

Tip: To quickly find the Workbench Native Application Administration portlet, select Title starts with in the Search by field and specify Rich Client in the Search field.
6. Click the copy portlet icon to create a copy.
7. Name your copy of the portlet as appropriate and click **OK**.
8. Place the new copy on the Workbench Contribution Page.
9. Click the Portal down-arrow and select **Configure** from the drop down menu.
10. Configure the native application settings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Default label for the application. If an appropriate localizable value is not set for the label, this value is used. The label is what is displayed in the launcher as the application title. The ampersand (&amp;) and double-quote (&quot;&quot;) characters are not allowed to be part of the label string.</td>
</tr>
<tr>
<td>Launcher path</td>
<td>Defines the location inside the launcher, where the application should appear. The launcher path is a hierarchical delimitation of the parent folders of the launcher item, separated by backslashes. Each element in the path corresponds to the name of a folder to be displayed in the launcher, leading from the root to the final launcher item.</td>
</tr>
<tr>
<td>Launch automatically at start up</td>
<td>If true, the application is started immediately after the launcher widget is loaded. If multiple applications are set to auto launch, there is no well defined order of execution.</td>
</tr>
<tr>
<td>Allow Multiple Instances</td>
<td>If true, then any attempt to launch an application more than once will result in multiple instances of the same application be opened. If false, then any attempt to launch more than one instance of the application will bring the existing application instance window to front, if it is still open.</td>
</tr>
<tr>
<td>Command</td>
<td>The command used to launch the application. A separate command is specified for running under windows and Linux. The command includes the path to the executable image plus any additional parameters.</td>
</tr>
<tr>
<td>Working Directory</td>
<td>The working directory of the application, after startup.</td>
</tr>
<tr>
<td>Environment</td>
<td>The environment variable definitions available to the application.</td>
</tr>
<tr>
<td>Environment Mode</td>
<td>Controls how additional environment variable definitions will be merged with any existing environment variables:</td>
</tr>
<tr>
<td>Append</td>
<td>If the original environment contains a definition for var1 = A, and the new environment contains a definition for var1 = B, then the final environment contains the definition Var1 = A;B</td>
</tr>
<tr>
<td>Prepend</td>
<td>If the original environment contains a definition for var1 = A, and the new environment contains a definition for var1 = B, then the final environment contains the definition Var1 = B;A</td>
</tr>
<tr>
<td>Override</td>
<td>If the original environment contains a definition for var1 = A, and the new environment contains a definition for var1 = B, then the final environment contains the definition Var1 = B;</td>
</tr>
<tr>
<td>Replace</td>
<td>The new environment will completely replace the old environment. No merge is done.</td>
</tr>
</tbody>
</table>
Table 47. Native application settings (continued)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set localizable values</td>
<td>The link will load a page similar to the following mockup, and allow the administrator to configure the localizable values for tooltip, launcher path, and label. Each row in the localization table defines a supported language, where each column displays a field that supports localized values. The list of supported languages depends on what languages the Portal installation supports. To enable your Portal installation for additional language support, refer to your IBM WebSphere Portal manual. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the Editing localized values screen, enter localized values for the selected language.</td>
</tr>
</tbody>
</table>

11. Assign appropriate access permissions to the portlet. On the Administration navigation panel, expand Access and click Resource Permissions.

12. In the Resource Types table, click Portlets.

13. Locate the copy of the portlet created above, and click on the access permissions icon to assign appropriate permissions.

A new instance of the Native Application Administration portlet is created and is configured to represent the native application that appears on the launcher.

**Contributing the screen lock function to the workbench:**

To enable users to lock up their screen, place the Workbench Screen Lock Administration portlet on the Workbench Contribution page.

Use the Workbench Screen Lock Administration portlet to add a toolbar button, menubar item, or status bar contribution to the workbench. The Workbench Screen Lock Administration portlet allows users to lock their screen from within the workbench.

1. Place the Workbench Screen Lock Administration portlet on the Workbench Contribution page and enter Configure mode to configure the settings.

2. Configure the settings for the Workbench Screen Lock Administration portlet. Select the location on the workbench where the screen lock function is to appear:
   - Show as a workbench menubar item
   - Show as an icon on the workbench status bar
   - Show as an icon on the workbench toolbar

   Check each area where the screen lock function should appear.

3. Click **Set localizable values** to set localizable values for the Tooltip and Label fields.

4. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the follow-on screen (Editing localizable values), enter localized values for the selected language. Enter the localized value for each field and click **OK**.

5. Click **OK** again to return to the configure screen.

6. Click **Set advanced options** to set the advanced options associated with this portlet.

7. Assign appropriate access permissions to the portlet. In the Administration navigation panel, expand Access and click Resource Permissions. In the resource type table, click Portlets. Locate the Workbench Screen Lock Administration portlet and click the access permissions icon to assign appropriate permissions.

A new instance of the Workbench Screen Lock Administration portlet is created and is configured on a WebSphere Portal page.

**Contributing the change password function to the workbench:**
To enable users to manage passwords, place the Workbench Change Password Administration portlet on the workbench contribution page.

Use the Workbench Change Password Administration portlet to add a toolbar button or preference page to the workbench. The Workbench Change Password Administration portlet allows users to change their operating system password from within the workbench, optionally synchronizing their main credential store password as well, if Single Sign On is active.

Follow these steps to contribute the change password function to the workbench.
1. Place the Workbench Change Password Administration portlet on the workbench contribution page and enter Configure mode to configure the settings.
2. Configure the settings for the Workbench Change Password Administration portlet. Select the location on the workbench where the change password function is to appear:
   • Show as an icon on the workbench toolbar
   • Show as a workbench preference page

   Check each area where the change password function should appear.
3. Click **Set localizable values** to set localizable values for the Tooltip and Label fields.
4. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the follow-on screen (Editing localizable values), enter localized values for the selected language. Enter the localized value for each field and click **OK**.
5. Click **OK** again to return to the configure screen.
6. Click **Set advanced options** to set the advanced options associated with this portlet.
7. Assign appropriate access permissions to the portlet. In the Administration navigation panel, expand **Access** and click **Resource Permissions**. In the resource type table, click **Portlets**.
8. Locate the Workbench Change Password Administration portlet and click the access permissions icon to assign appropriate permissions.

The Workbench Change Password Administration portlet is placed on a WebSphere Portal page and configured with the correct settings.

**Contributing the change locale function to the workbench:**

Add the change locale function to enable users to manage the operating system locale.

Use the Workbench Change Locale Administration portlet to add a toolbar button, preference page, or status bar contribution to the workbench.

**Note:** If your Lotus Expeditor Client platform is Windows-based platform, you must restart your computer for the locale change to take effect. If your platform is Linux-based, you must logoff and log back on for the locale change to take effect.

To add the change locale function, follow these steps:
1. Place the Workbench Change Locale Administration portlet on the Workbench Contribution page and enter Configure mode to configure the settings.
2. Configure the settings for the Workbench Change Locale Administration portlet. Select the location on the workbench where the change locale function is to appear:
   • Show as an icon on the workbench status bar
   • Show as an icon on the workbench toolbar
   • Show as a workbench preference page
3. Click **Set localizable values** to set localizable values for the Tooltip and Label fields.
4. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the follow-on screen (Editing localizable values), enter localized values for the selected language. Enter the localized value for each field and click OK.

5. Click OK again to return to the configure screen.

6. Click Set advanced options to set the advanced options associated with this portlet.

7. Assign appropriate access permissions to the portlet. In the Administration navigation panel, expand Access and click Resource Permissions. In the resource type table, click Portlets.

8. Locate the Workbench Change Locale Administration portlet and click the access permissions icon to assign appropriate permissions.

The Change Locale Administration portlet is configured on a WebSphere Portal page.

**Contributing the change keyboard function to the workbench:**

To enable users to change the operating system keyboard layout, place the Workbench Keyboard Layout Administration portlet on the Workbench Contribution page.

Use the Workbench Keyboard Layout Administration portlet to add a toolbar button or preference page to the workbench. The Workbench Keyboard Layout Administration portlet allows users to change their operating system keyboard layout from within the workbench.

1. Place the Workbench Keyboard Layout Administration portlet on the Workbench Contribution page and enter Configure mode to configure the settings.

2. Configure the settings for the Workbench Keyboard Layout Administration portlet. Select the location on the workbench where the change keyboard function is to appear:
   - Show as an icon on the workbench status bar
   - Show as an icon on the workbench toolbar
   - Show as a workbench preference page

Check each area where the change keyboard layout function should appear.

3. Click Set localizable values to set localizable values for the Tooltip and Label fields.

4. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the follow-on screen (Editing localizable values), enter localized values for the selected language. Enter the localized value for each field and click OK.

5. Click OK again to return to the configure screen.

6. Click Set advanced options to set the advanced options associated with this portlet.

7. Assign appropriate access permissions to the portlet. In the Administration navigation panel, expand Access and click Resource Permissions. In the resource type table, click Portlets. Locate the Workbench Change Keyboard Layout Administration portlet and click the access permissions icon to assign appropriate permissions.

A new instance of the Workbench Change Keyboard Administration portlet is created and is configured on a WebSphere Portal page.

**Contributing the Tasklist function to the workbench:**

Add the Tasklist function to enable users to view a list of running native tasks.

Use the Workbench Tasklist Administration portlet to add a toolbar button or status bar contribution to the workbench.

To add the tasklist function, follow these steps:

1. Place the Workbench Tasklist Administration portlet on the Workbench Contribution page and enter Configure mode to configure the settings.
2. Configure the settings for the Workbench Tasklist Administration portlet. Select the location on the workbench where the tasklist function is to appear:
   - Show as an item on the workbench status bar
   - Show as an icon on the workbench toolbar
3. Click **Set localizable values** to set localizable values for the Tooltip and Label fields.
4. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the follow-on screen (Editing localizable values), enter localized values for the selected language. Enter the localized value for each field and click **OK**.
5. Click **OK** again to return to the configure screen.
6. Click **Set advanced options** to set the advanced options associated with this portlet.
7. Assign appropriate access permissions to the portlet. In the Administration navigation panel, expand **Access** and click **Resource Permissions**. In the resource type table, click **Portlets**. Locate the Workbench Planned Connectivity Administration portlet and click the access permissions icon to assign appropriate permissions.

**Contributing the system shutdown function to the workbench:**

Use the system shutdown function portlet to add a toolbar button or preference page to the Lotus Expeditor Client workbench. The Workbench System Shutdown Administration portlet enables users to shutdown from within the workbench.

Use the Workbench System Shutdown Administration portlet to add a toolbar button or preference page to the workbench. The Workbench System Shutdown Administration portlet allows users to shutdown the system from within the workbench.

**Note:** The Restricted Workbench Service must be enabled on the client for the Workbench System Shutdown to be effective.

1. Place the Workbench System Shutdown Administration portlet on the Workbench Contribution page and enter Configure mode to configure the settings.
2. Configure the settings for the Workbench System Shutdown Administration portlet. Select the location on the workbench where the system shutdown function is to appear:
   - Show as an icon on the workbench status bar
   - Show as an icon on the workbench toolbar
   - Show as an item on the workbench menu bar

   Check each area where the shutdown function should appear.
3. Click **Set localizable values** to set localizable values for the Tooltip and Label fields.
4. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the follow-on screen (Editing localizable values), enter localized values for the selected language. Enter the localized value for each field and click **OK**.
5. Click **OK** again to return to the configure screen.
6. Click **Set advanced options** to set the advanced options associated with this portlet.
7. Assign appropriate access permissions to the portlet. In the Administration navigation panel, expand **Access** and click **Resource Permissions**. In the resource type table, click **Portlets**. Locate the Workbench System Shutdown Administration portlet and click the access permissions icon to assign appropriate permissions.

A new instance of the Workbench System Shutdown Administration portlet is created and is configured on a WebSphere Portal page.

**Related information**
Contributing the system logoff function to the workbench:

Add the system logoff function to enable users to logoff the system.

Use the Workbench System Logoff Administration portlet to add a toolbar button, menubar, or status bar contribution to the workbench.

Note: The Restricted Workbench Service must be enabled for the Workbench System Logoff Administration portlet to work.

To add the system logoff, follow these steps:
1. Place the Workbench System Logoff Administration portlet on the Workbench Contribution page and enter Configure mode to configure the settings.
2. Configure the settings for the Workbench System Logoff Administration portlet. Select the location on the workbench where the system logoff function is to appear:
   - Show as an icon on the workbench menubar
   - Show as an icon on the workbench toolbar
   - Show as an item on the workbench status bar
3. Click Set localizable values to set localizable values for the Tooltip and Label fields.
4. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the follow-on screen (Editing localizable values), enter localized values for the selected language. Enter the localized value for each field and click OK.
5. Click OK again to return to the configure screen.
6. Click Set advanced options to set the advanced options associated with this portlet.
7. Assign appropriate access permissions to the portlet. In the Administration navigation panel, expand Access and click Resource Permissions. In the resource type table, click Portlets. Locate the Workbench System Logoff Administration portlet and click the access permissions icon to assign appropriate permissions.

A new instance of the shutdown admin portlet is created and is configured on a WebSphere Portal page.

Related information

Restricted Workbench Service on page 3
Enable the Restricted Workbench Service to provide a restricted environment in which all Lotus Expeditor Client users are limited to the applications and operating system services that you, as administrator, have configured.

Contributing the status bar clock to the workbench:

Contributing the Workbench Clock Administration portlet to the workbench enables users to add a digital clock to the status bar of the workbench.

To add the status bar clock, follow these steps:
1. Place the Workbench Clock Administration portlet on the Workbench Contribution page and enter Configure mode to configure the settings.
2. Configure the settings for the Workbench Clock Administration portlet.
3. Click Set localizable values to set localizable values for the Tooltip and Label fields.
4. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the follow-on screen (Editing localizable values), enter localized values for the selected language. Enter the localized value for each field and click OK.

5. Click OK again to return to the configure screen.

6. Click Set advanced options to set the advanced options associated with this portlet.

7. Assign appropriate access permissions to the portlet. In the Administration navigation panel, expand Access and click Resource Permissions. In the resource type table, click Portlets.

8. Locate the Workbench Clock Administration portlet and click the access permissions icon to assign appropriate permissions.

A new instance of the clock admin portlet is created and is configured on a WebSphere Portal page.

### Contributing a custom dynamic eclipse extension to the workbench:

To enable users to have the ability to contribute any valid eclipse extension, place the Workbench Custom Eclipse Extension Administration portlet on the Workbench Contribution page.

To contribute a custom Eclipse extension, follow these steps:

1. Place an instance of the Workbench Custom Eclipse Extension Administration portlet on the Workbench Contribution page and enter Configure mode to configure the settings.

2. Configure the settings for the Workbench Custom Eclipse Extension Administration portlet. To define a valid eclipse extension, supply the following:

   **Host Bundle ID**
   - ID of the plug-in where the extension will be contributed.

   **Eclipse Extension Definition**
   - XML of the eclipse extension being defined. Multiple extensions can be defined using a single instance of the portlet, assuming all extensions are contributed to the same plug-in.

3. Click Set localizable values to set localizable values for the Tooltip and Label fields.

   **Note:** The Set localizable values link will only appear if the Eclipse extension definition contains a localizable variable. The extension definition contains localizable variables if it contains tokens like `%variable%`.

4. To enter localized values for a supported language, click the edit icon in the row corresponding to that language. In the follow-on screen (Editing localizable values), enter localized values for the selected language. Enter the localized value for each field and click OK.

5. Click OK again to return to the configure screen.

6. Click Set advanced options to set the advanced options associated with this portlet.

7. Assign appropriate access permissions to the portlet. In the Administration navigation panel, expand Access and click Resource Permissions. In the resource type table, click Portlets. Locate the Workbench Custom Eclipse Extension Administration portlet and click the access permissions icon to assign appropriate permissions.

A new instance of the Workbench Custom Eclipse Extension Administration portlet is created and is configured on a WebSphere Portal page.

### Assigning access control to launcher items:

Assigning access control to the launcher items means assigning access control to the portlets on the Portal server that represents each launcher item. It is the same as assigning access control to any portlet.

Using the Rich Client WSRP Viewer Enablement Portlet

Use the Rich Client WSRP Viewer Enablement Portlet to configure a WSRP portlet for the Lotus Expeditor Client platform.

The Rich Client WSRP Viewer Enablement Portlet is a proxy portlet for a WSRP portlet that stores the WSRP metadata (the WSRP portlet handle and producer information). After the Rich Client WSRP Viewer Enablement Portlet is configured and deployed in a composite application, the composite application can present a portlet viewer in the Lotus Expeditor Client platform that shows the content of the remote WSRP portlet.

WSRP is an OASIS standard that defines interactive, presentation-oriented Web services between WSRP-enabled applications, such as the Portal server or the WSRP runtime within the Lotus Expeditor Client. The WSRP runtime retrieves portlet content from remote portlets and processes user interaction.

This topic assumes that you are already familiar with the WSRP specification. For more details about the OASIS WSRP specification, refer to the OASIS WSRP Standard Web site: [http://oasis-open.org/committees/wsrp](http://oasis-open.org/committees/wsrp)

If you want to deploy a portlet viewer for WSRP runtime in the Portal-administered client environment, you need to configure and deploy a Rich Client WSRP Viewer Enablement Portlet. Use the Rich Client WSRP Viewer Enablement Portlet to search remote portlets that provide WSRP services and share their WSRP-related metadata in a Rich Client view. Additional data needed by the portlet viewer for WSRP runtime is configured using the Rich Client Layout Administration Portlet, which is described in “Using the Rich Client Layout Administration Portlet” on page 209. Once you deploy a Rich Client WSRP Viewer Enablement Portlet in a composite application and configure it for a remote portlet, the Portal-administered client will present the portlet viewer for WSRP runtime showing the contents of the remote portlet.

Note: Deploy only WSRP-compliant portlets with the Lotus Expeditor Client. Even though you can specify a portlet in the Portal server’s WSRP interface or command line, the portlet is not considered WSRP-compliant. Refer to the WSRP Standard Web site for information about WSRP-compliancy. Before deploying the WSRP functionality in the Lotus Expeditor Client, it’s best to try to consume the WSRP service provider in the management server.

Related information

“Configuring the Rich Client WSRP Viewer Enablement Portlet”

Before you can deploy the Rich Client WSRP Viewer Enablement Portlet, it must be configured.

“Deploying a Rich Client WSRP Viewer Enablement Portlet” on page 233

Deploy a WSRP client component enablement portlet to allow an existing consumer-configured portlet instance to be accessed by your Lotus Expeditor Client through WSRP.

Configuring the Rich Client WSRP Viewer Enablement Portlet:

Before you can deploy the Rich Client WSRP Viewer Enablement Portlet, it must be configured.

To configure the Rich Client WSRP Viewer Enablement Portlet:

1. Make sure the Rich Client WSRP Viewer Enablement Portlet is properly installed on WebSphere Portal.
2. Log on to WebSphere Portal using administrator’s credentials.
3. Click Administration under Product Links.
5. Locate the portlet named Rich Client WSRP Viewer Enablement Portlet.

   Tip: To quickly find the Rich Client WSRP Viewer Enablement Portlet, select Title contains in the Search by field and specify Rich Client WSRP Viewer Enablement in the Search field.
6. Click the copy portlet icon to create a copy.
7. Name your copy of the portlet as appropriate and click **OK**.
8. Place the new copy on the appropriate perspective page.
9. Click the Portal down-arrow and select **Configure** from the drop down menu.

    **Note:** If you are using WebSphere Portal as the WSRP producer, refer to **Using WSRP services with WebSphere Portal → Producer tasks → WSRP service description** in the WebSphere Portal information center.

11. Enable the security option for this producer. Make sure you are familiar with the security configuration of the WSRP producer you specified. Select from the following security options:

    **No Security**
    WSRP producer does not have any security configuration.

    **LTPA**
    The WSRP producer is using LTPA token for authentication. If the WSRP producer is not in the Portal server that hosts the Rich Client WSRP Enablement Portlet, you should configure single sign-on between these two Portal servers.

    **Note:**
    - For the security configuration for Portal, refer to **Using WSRP services with WebSphere Portal → Security considerations for using WSRP with WebSphere Portal** in the WebSphere Portal information center.
    - After configuring single sign-on for the WSRP producer server and the Portal server that hosts the Rich Client WSRP Enablement Portlet, you cannot use IP to set up other configurations to identify these servers.

    **Username Token**
    WSRP producer is using Username token for authentication. If you select this option, you need to specify an available account on this Portal server in the Credential Vault portlet, using the URL in step 10 as the resource name. Note that the Credential Vault portlet will also need a "Slot Name", it is suggested to set the same name for both the "resource name" and the "slot name".

12. Click **OK** to go to the next page.
13. Select the portlet you want to deploy from the list of WSRP-enabled portlets for this producer. The next page is displayed.
14. Specify the client-side clone options. Select from the following options:

    **Don’t need a clone**
    All of the WSRP portlet viewers on the client-side will share one remote portlet instance.

    **Need a clone**
    Every WSRP portlet viewer will clone a new portlet instance in the WSRP producer.

    **Note:** If the WSRP producer is the WebSphere Portal server, select **Don’t need a clone**; otherwise, select **Need a clone**.

15. Click **Submit** to save the changes and return to the view mode.

The Rich Client WSRP Enablement Portlet is configured and ready to be deployed.

**Deploying a Rich Client WSRP Viewer Enablement Portlet:**

Deploy a WSRP client component enablement portlet to allow an existing consumer-configured portlet instance to be accessed by your Lotus Expeditor Client through WSRP.
Before you can deploy a WSRP enablement viewer, make sure the WSRP client component enablement portlet is installed on the Portal server and that the WSRP feature is installed on the Lotus Expeditor Client.

1. Complete the steps in “Configuring the Rich Client WSRP Viewer Enablement Portlet” on page 232.

Once completed, you can see and interact with the portlet instance using the WSRP viewer.

Registering
This section describes Portal registering.

Defining Portal access control levels
The level of access that a user has to portlets and pages hosted by WebSphere Portal must be set using the WebSphere Portal Administration pages.

Applications that you build to run on the client must specify that all authenticated WebSphere Portal users can use the portlets and pages that make up your application for the application to be made available in the client.

To set the access level for portlets and pages to all authenticated WebSphere Portal users:
1. From the WebSphere Portal Administration pages, click Access → Resource Permissions → Pages.
2. Search for pages with "WMC" to display the page you created.
3. Click the Assign Access key icon of the page you created.
4. Click the Edit Role icon for the User role, and then click Add.
5. Check all authenticated portal users, and then click OK. Click Done, then click OK to return to the page search results. Click Done to return to the list of resource types. Click Portlets, and then search for portlets with "WMC" to return the portlets you created.
6. For each portlet, set the User role to all authenticated portal users.

Specifying MyClient applications
Applications you specify in the MyClient category are automatically installed and updated when Lotus Expeditor Client is administered by the Portal server.

Lotus Expeditor Client loads the applications specified in the catalog that are in the MyClient category.

Managing Rich Client Preferences using the Policy Type Editor for Eclipse Preferences
The Resource Policies portlet is a WebSphere Portal administrative tool that allows you to manage sets of key value pairs.

The Lotus Expeditor Client provides an extension to the Resource Policies portlet called the Policy Type Editor for Eclipse Preferences through which Portal-administered client can be managed. The Policy Type Editor for Eclipse Preferences integrates seamlessly with other Resource Policies extensions.

Preinstalled preference sets
The items in bold show on the GUI pages as links or form-entry labels. The rest of the information is for explanation.

Client connectivity preferences
   Real name: ClientConnectivityPreference
Description: Configures the network layer on the rich client.

Preferences:
- **Time interval (in minutes) for polling network availability**
  - Key: com.ibm.rcp.net.defaults/local_monitor_interval
  - Description: interval at which the network framework service should monitor local network activity
  - Initial value: 5
  - Constraints: integer, 1 to 2440

**Client managed settings**
Real name: ManagedSettings
Description: Configures the managed settings framework on the rich client

Preferences: **Time interval (in minutes) for updating policies on the client**
- Key: com.ibm.rcp.managedsettings/UpdateIntervalInMins
- Description: Interval at which Managed Settings are updated from the back-end settings system
- Initial value: 720 Constraints - integer > 0

**Client portlet viewer settings**
Real name: PortletEditModeAllow
Description: Configures the portlet viewer on the rich client

Preferences: **Allow Edit mode on the client portlet viewer**
- Key: com.ibm.rcp.portletviewer/PortletEditModeAllow
- Description: Used to determine if the Edit mode icon or selection is displayed
- Initial value: false
- Constraints: Boolean (true, false)

**Modifying a value in a policy type**
The policy type contains the default values. You can update the default value for a preference in the policy type.

Follow these steps to modify a value in a policy type.
1. Log in to WebSphere Portal as an administrator.
2. Click **Administration** in the Portal toolbar.
3. Click **Resource Policies**. A list of existing Policy Types is displayed.
4. Locate the policy type you want to change.
5. Open the pop-up menu and select **Edit Policy**.
6. Make the necessary changes to the values and click **OK**.

The policy type is updated with new values.

**Creating a child policy for a policy type**
Child policies can be used with personalization to allow different settings for different types of users. You can create a hierarchical tree of policies for each policy type.

Use policies to specify and apply common and specialized settings that determine the how Portal resources function for different classes of users.

A policy is a collection of settings. Some policies influence the behavior of a Portal resources and the experience that users will have when working with those resources. Other policies influence behavior of resources on a client. Policies simplify the management of resources because the policy settings for a resource type can control the behavior of the resource for different classes of users. For sites that
comprise large numbers of users and resources, specifying and applying policies to resources eliminates much time-consuming effort that would otherwise require administrators to manage a vast array of discrete settings for users and resources.

To create a child policy, you must first assign a policy rule to the parent. The top-most parent is the policy type. You assign policy rules to a policy type using WebSphere Portal’s Personalization and Resource Policies functions. See the WebSphere Portal information center for information about managing Portal resources with policies.

Follow these steps to create a child policy.
1. Log in to WebSphere Portal as an administrator.
2. Click Administration in the Portal toolbar.
3. Click Resource Policies. A list of existing policy types is displayed.
4. Locate the policy type you want to change.
5. Click the policy type. All the existing child policies for this policy type are displayed.
   - If you selected a personalization rule, the policy rule’s name is displayed. If no policy rule was selected, use the pop-up menu for the policy rule to create a new Personalization rule or select from an existing personalization rule.
   - The New Policy button is disabled until a rule is selected.
6. Click the New Policy button.
7. Enter a title and a description for this policy, and select a condition for this policy.
8. Change the required values and click OK.

The child policy is created and you are returned to the Resource Policies window.

**Editing values in a child policy**
You can edit a child policy.

Follow these steps to edit a child policy. The Resource Policies window allows you to navigate through the hierarchical tree of child policies to locate the policy you want to edit.
1. Log in to WebSphere Portal as an administrator.
2. Click Administration in the Portal toolbar.
3. Click Resource Policies. A list of existing policy types is displayed.
4. Locate the policy type you want to update.
5. Click the policy type. All first-level child policies are displayed.
6. If necessary, click on a child policy to navigate deeper into the tree of child policies.
7. Once you have located the child policy you want to change, use the context menu to select Edit Policy.
8. Change the required values and click OK.

The child policy is updated with the new values and you are returned to the Resource Policies window.

**Installing a new preference set**
Most of the preference sets you use are automatically installed with the Lotus Expeditor Client. If additional preference sets are required, you can install them using the Portlet’s import capability.

When you import a preference set, a new policy type containing the default values is created.

Creating a new policy value set for your custom plug-ins or the plug-ins provided with the Lotus Expeditor Client allows you to centrally manage Eclipse preferences from the server. See Developing Applications for Lotus Expeditor for instructions on creating new preference sets.
**Sample scenario:** Your company has service centers in different regions of the country and you want to automate which service center the employees in your company should access based on where they live. You can create your own plug-in that uses an Eclipse preference to contain a value based on their location:

com.mycompany.serviceplugin/MYSERVICECENTER=Raleigh

You can write an XML template containing this preference. You would install the template, along with a personalization rule based on the employee’s location that is assigned as the policy type. The final step would be to create child policies based on the personalization rule. This would allow the service center to be automatically assigned.

Follow these steps to import a new preference set on the server. These steps assume that you have been provided an XML template in the Policy Type Editor for Preferences schema.

1. Use a browser to access **Administration ➤ Resource Policies ➤ Rich Client Templates**.
2. On the portlet drop-down menu in the Portlet header bar, click **Configure** to enter configuration mode.
3. Click **Browse** to navigate through the local file system and select the XML template file.
4. Select the file, and click **Import**. The XML template is validated and saved in the Portlet settings. The policy type is created.

You can now access the policy type using **Administration ➤ Resource Policies**.

**Note:**

- If your new policy type does not appear, you can either click on one of the policy types and click on the **Policy Types** link at the top of the page or log off the Portal server and log back on.
- To delete a user-defined template, you must use Policy Editor XMLAccess Scripts, which are described in “Sample Policy XML configuration files” in the **WebSphere Portal information center**. It is recommended that you remove the policy type if you uninstall a user-defined template. If you uninstall the Policy Type Editor, templates are deleted but the policy types are not deleted. You cannot edit the policy type unless you reload the template even after re-installing the Policy Type Editor. It is recommended that you save templates in another location.

**Managing properties**
There are no specific tasks provided by the Portal Server for managing properties files. You can affect the contents of properties files by using install handlers within the features that you deploy.

See [*Using the global install handler* on page 15] for more information about platform supplied install handlers.

**Managing using another management system**
The client platform provides two models of platform and application management: using the Enterprise Management Agent and the Device Manager server, and using the Portal Server to control application layout and provisioning of the applications to the client. If you choose not to use these capabilities, you can use another management system to manage the client platform.

The critical element of managing the platform without using Device Manager server or the Portal Managed Client capabilities is that all applications must be installed as features. This requires that the applications be packaged into a feature, and distributed as part of an update site.

**Configuring the update manager policy**
Refer to the Eclipse documentation for information on setting the policy and configuring automatic updates.
Managing preferences
The Device Manager Server and the Portal Managed client capabilities run within the context of the platform, so have access to the APIs necessary to update preference values. If you are not using these built in capabilities, changing preference values is more difficult.

You can update the default preference values by changing the plugin_customization.ini file, but if you want to perform updates on the current values, you will need to call APIs to make the changes.

You can use install handlers to set initial default values, but if you need to query current values, or perhaps make updates to existing values, you will need to provide your own mechanism to call the APIs.

You can also use the Managed Settings provider infrastructure to implement your own preferences settings capabilities.

While the preferences values are stored in the file system on the local machine, it is not recommended that the files be changed outside of the APIs.

Managing properties
The platform provides a number of configuration files, as defined in the section on Configuring the platform.

The preferred model for updating the properties files that are part of the client platform is to add the global install handler to any feature that needs to update the configuration files. However, you are not limited to this as a solution for modifying the properties files. If you should choose to use your own management system to modify these files, please note that these are formatted as Java properties files, and there are specific rules for the formatting of these files. Refer to documentation on the java.util.Properties class for information on these restrictions.

Using the provisioning interfaces to manage the platform
This section explains how to use the available provisioning interfaces to manage the client platform.

If the Device Manager server or the Portal server are used to manage the platform, most software distribution tasks are already handled. If another management system is used to manage the client platform, it must use the available provisioning interfaces to effect changes to the system. There are primarily three options for using the provisioning interfaces:

- Directly update the provisioning manifest and related properties, and restart the platform.
- Directly call the Provisioning Application to perform changes to the platform.
- Use Provisioning APIs from within the platform to perform changes to the platform.

These methods provide access to the provisioning system that is part of the platform to ensure that features and plug-ins are installed and enabled correctly.

---

Attention: 6.1.2 The Eclipse standaloneUpdate command is disabled in Lotus Expeditor 6.1.2. It is recommended that you use provisioning interfaces to manage the platform.

Updating the Provisioning Manifest to change the platform
The provisioning application is initially run during the installation phase of Lotus Expeditor. The Provisioning Application can be rerun at any point in time to effect updates to the platform.
To run the Provisioning Application again, update the provision.manifest.version property in the rcplauncher.properties file to a value greater than current, and update the manifest file contents. If the current value present in rcpinstall.properties is less than the value of the provision.manifest.version specified in the rcplauncher.properties, the launcher will automatically start the provisioning application to process the manifest.

Alternatively, all the feature changes can be supplied in a <product install dir>/rcp/deploy/install.update.xml manifest file and run the provisioning update command to merge the changes. Once the manifest has been processed, Expeditor must restart to display the selected application or personality, or both.

The format of the provisioning manifest is referenced in “Provisioning manifest DTD” on page 165.

**Using the Provisioning application**

If provisioning steps need to be explicitly performed, such as by another management system, the Provisioning application can be invoked using the command line.

The syntax for the Provisioning application is as follows:

```
rcplauncher
  -application
com.ibm.rcp.provisioning.application.ProvisioningApplication
  -provisioningOperation { install | update | uninstall | provision filename }
  -provisioningStatusLog filename
```

The Provisioning application reads the rcp/rcplauncher.properties file to obtain the provision.manifest and provision.manifest.version properties. The Provisioning application parses the manifest and processes all features present in the file. Boundaries between install features are ignored by the Provisioning application.

Upon successful (or partially successful) install, the provision.manifest.version property is updated in the rcp/rcpinstall.properties file.

After the processing of the arguments, the system property provisioningOperation is set to the operation currently being processed.

The **install** operation results in a parsing of the provisioning manifest. All features present in the file are compared to the current features present on the platform, and installed and enabled if needed.

The **uninstall** operation is called twice. The first time that it is called, it will disable all of the features on the platform, except for those required to run the core platform and provisioning. The second time that it is called, it will uninstall all of the disabled features on the platform.

The **update** operation processes the provisioning manifest. If a file install.update.xml is present in the same directory as the provisioning manifest, then the contents will be merged prior to processing the provisioning manifest to perform any updates.

The **provision** operation processes the provisioning manifest supplied as an argument on the command and processes the features contained within the provisioning manifest. Contents of the supplied provisioning manifest file are not merged with the version defined by the platform, and there are no changes to the provision.manifest.version property in the rcplauncher.properties file.

**Note:** Ensure that you do not end the provisioning manifest path with a backslash "\" because the VM will interpret this as an escape character, which results in an error.

**Using the Provisioning application with DCommands:**
Provisioning operations can also be initiated if the platform is already running. Through the use of DCommands, command arguments can be passed into a receiving class inside of the running platform.

To start the platform such that either the DCommands are passed through to a running instance, or a new instance is launched to process the arguments, use the following syntax to launch:

```
rcplauncher
  -com.ibm.brtrcp.provisioning -provisioningcommand -provisioningoperation command
  -provisioningcommand com.ibm.rcp.provisioning.application.ProvisioningApplication
  -provisioningOperation -provisioningCommand -provisioningOperation
  -provisioningStatusLog file
```

This will result in one of two results:

- The existing platform (currently running the workbench) is passed the provisioning arguments, and the provisioning occurs inside of the existing platform. When provisioning is complete, the existing platform will continue to run until the user currently using the platform decides to shut down.

- A new instance of the platform will start using the Provisioning application and the provisioning will occur using this new instance of the platform. When provisioning is complete, the platform will exit.

**Using the Provisioning application for multiuser configurations:**

In a multiuser configuration, use the Provisioning application to maintain the existing platform.

The Provisioning application is primarily intended for the deployer or administrator to use to update the main installation of features. The provisioning application, when used to do maintenance to the existing platform, will install, enable, disable, and remove features from the platform. To cause the other users on the platform to be able to access the changes, then the provisioning manifest should be updated, and the provisioning.manifest.version property should be updated. If these changes are not made, existing workspaces will continue to use their current configuration.

By using the mergemanifest command in the global install handler, the preferred model would be to distribute a new feature containing additional or updated provisioning manifest items that would be merged with the existing manifest. This will result in all users upgrading to the latest configuration at next launch of their workspace. (And all new workspaces being immediately set up with the latest feature configuration).

Users making use of the Provisioning application are only going to update their local application list (present in their workspace).

If a management system is intending to call the Provisioning application, then it will need to do so in the context of an administrator with rights to modify the existing platform configuration.

**Using the Provisioning APIs**

Use the Provisioning APIs to ensure correct and consistent installation of features and successful platform operation.

Use the Provisioning APIs to gain access to the provisioning system from other plug-ins running within the platform. In the event that applications running within the platform desire to make changes to the platform, such as by requesting the installation of new features and plug-ins, your applications should use the APIs provided rather than the Update APIs provided by Eclipse. By using the Provisioning APIs, it will ensure correct consistent installation of features and continued successful operation of the platform.

The general programming model for using the Provisioning APIs is to obtain an instance of the ProvisioningService from the ProvisioningServiceFactory, and then invoke the methods provided by the service to perform changes. The methods provided operate either on specific ProvisioningRequest objects, or on a provisioning manifest supplied to the service. For more information, see the Provisioning Javadoc in *Developing Applications for Lotus Expeditor*.
Notices

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

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

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department LZMS
Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM’s suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

COPYRIGHT LICENSE:

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

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2004, 2007 All rights reserved.
If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**Trademarks**

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

AIX
DB2
Domino
Everyplace
IBM
Lotus
Lotus Notes
Rational
Sametime
System p
Tivoli
WebSphere

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

JavaScript is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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